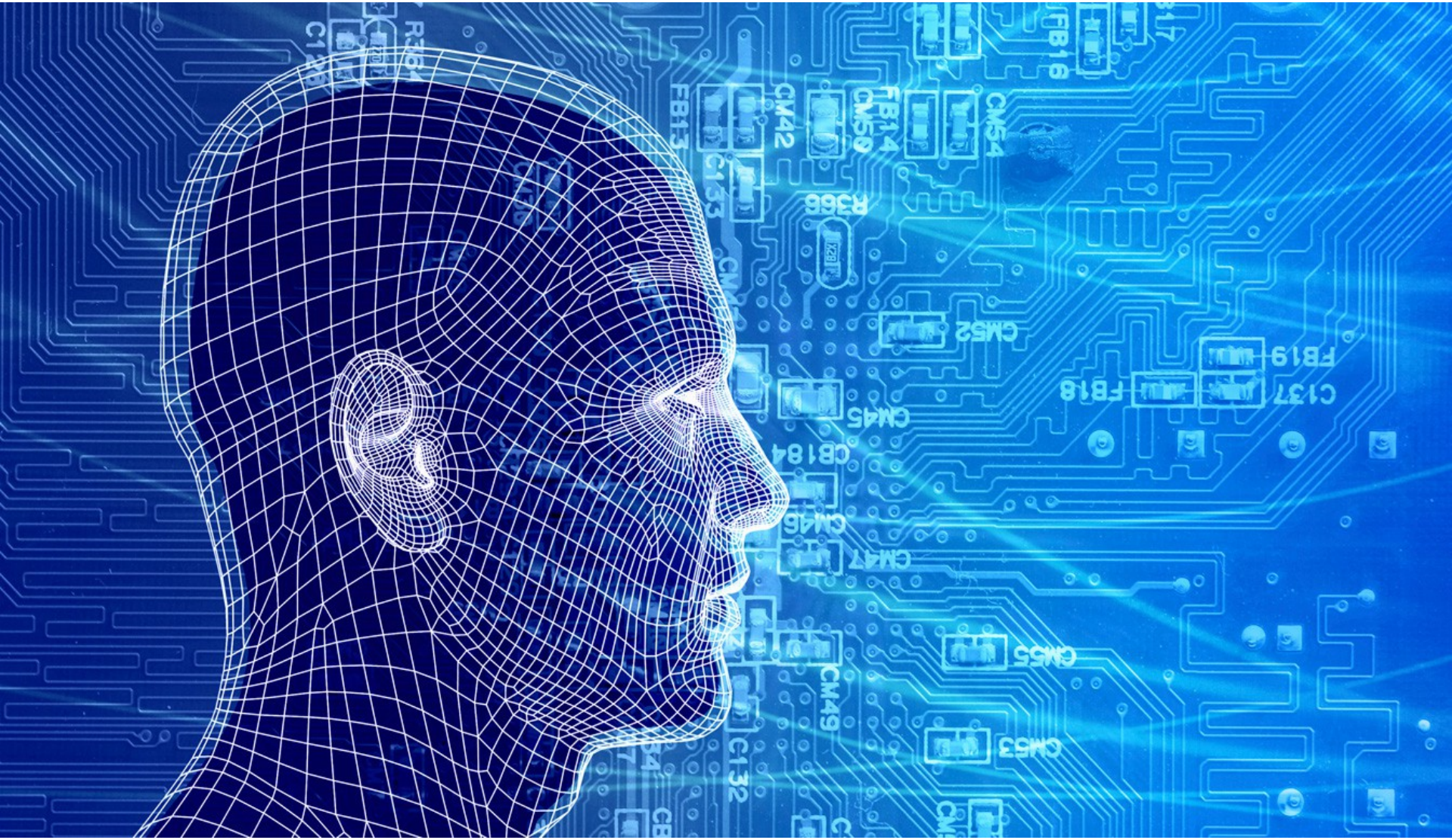
Can machines think like human beings and beyond!



UnSupervised Learning

Supervised learning vs. unsupervised learning

Supervised learning: discover patterns in the data that relate data attributes with a target (class) attribute.

- These patterns are then utilized to predict the values of the target attribute in future data instances.

Unsupervised learning: The data have no target attribute.

- We want to explore the data to find some intrinsic structures in them.

Clustering

Clustering is a technique for finding similarity groups in data, called **clusters**. I.e.,

- it groups data instances that are similar to (near) each other in one cluster and data instances that are very different (far away) from each other into different clusters.

Clustering is often called an **unsupervised learning** task as no class values denoting an *a priori* grouping of the data instances are given, which is the case in supervised learning.

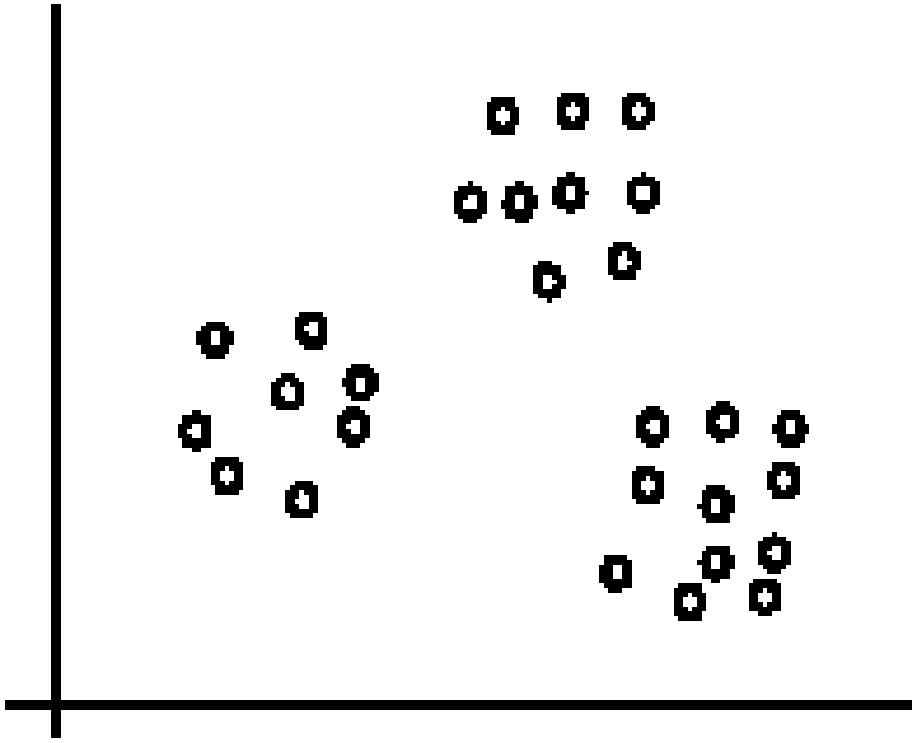
Due to historical reasons, clustering is often considered synonymous with unsupervised learning.

- In fact, association rule mining is also unsupervised

This chapter focuses on clustering.

An illustration

The data set has three natural groups of data points, i.e., 3 natural clusters.



What is clustering for?

Let us see some real-life examples

Example 1: groups people of similar sizes together to make "small", "medium" and "large" T-Shirts.

- Tailor-made for each person: too expensive
- One-size-fits-all: does not fit all.

Example 2: In marketing, segment customers according to their similarities

- To do targeted marketing.

What is clustering for? (cont…)

Example 3: Given a collection of text documents, we want to organize them according to their content similarities,

- To produce a topic hierarchy

In fact, clustering is one of the most utilized data mining techniques.

- It has a long history, and used in almost every field, e.g., medicine, psychology, botany, sociology, biology, archeology, marketing, insurance, libraries, etc.
- In recent years, due to the rapid increase of online documents, text clustering becomes important.

Aspects of clustering

A clustering algorithm

- Partitional clustering
- Hierarchical clustering
- …

A distance (similarity, or dissimilarity) function

Clustering quality

- Inter-clusters distance ! maximized
- Intra-clusters distance ! minimized

The quality of a clustering result depends on the algorithm, the distance function, and the application.

Example : clustering genes

E. Furlong *et al*., Patterns of Gene Expression During *Drosophila* Development, Science 293: 1629-33, 2001.

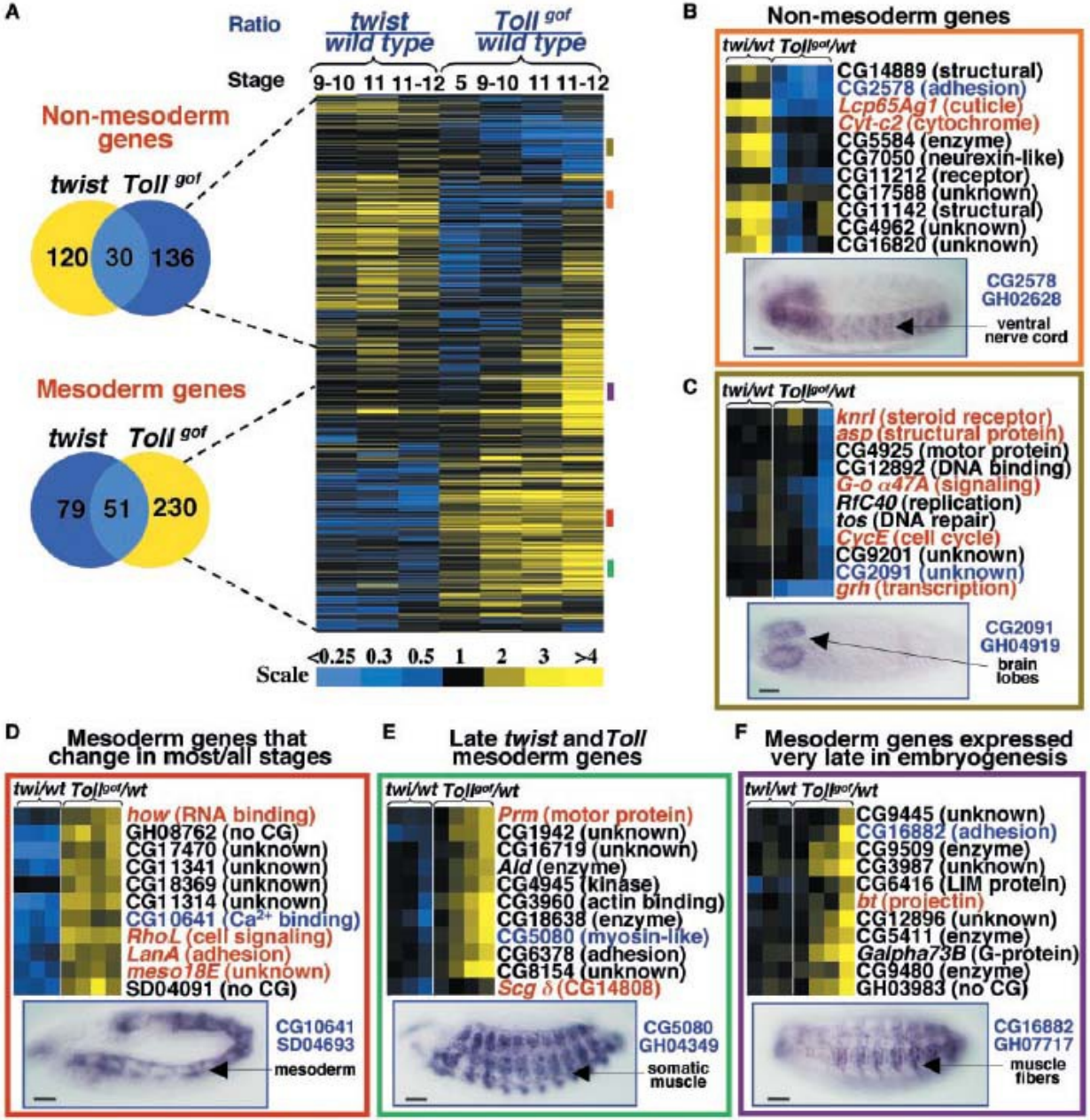
Use clustering to look for patterns of gene expression change in wild-type vs. mutants

Collect data on gene expression in *Drosophila* wild-type and mutants (*twist* and *Toll*) at three stages of development

twist is critical in mesoderm and subsequent muscle development; mutants have no mesoderm

Toll mutants over-express *twist*

Take ratio of mutant over wt expression levels at corresponding stages


A
Ratio
twist / wild type
Toll gof / wild type
Stage
9-10 11 11-12 5 9-10 11 11-12
Non-mesoderm genes
twist Toll gof
120 30 136
Mesoderm genes
twist Toll gof
79 51 230
<0.25 0.3 0.5 1 2 3 >4
Scale
B
Non-mesoderm genes
twi/wt Toll gof/wt
CG14889 (structural)
CG2578 (adhesion)
Lcp65Ag1 (cuticle)
Cyt-c2 (cytochrome)
CG5584 (enzyme)
CG7050 (neurexin-like)
CG11212 (receptor)
CG17588 (unknown)
CG11142 (structural)
CG4962 (unknown)
CG16820 (unknown)
CG2578
GH02628
ventral nerve cord
C
twi/wt Toll gof/wt
knrl (steroid receptor)
asp (structural protein)
CG4925 (motor protein)
CG12892 (DNA binding)
G-o α47A (signaling)
RfC40 (replication)
tos (DNA repair)
CycE (cell cycle)
CG9201 (unknown)
CG2091 (unknown)
grh (transcription)
CG2091
GH04919
brain lobes
D
Mesoderm genes that change in most/all stages
twi/wt Toll gof/wt
how (RNA binding)
GH08762 (no CG)
CG17470 (unknown)
CG11341 (unknown)
CG18369 (unknown)
CG11314 (unknown)
CG10641 (Ca2+ binding)
RhoL (cell signaling)
LanA (adhesion)
meso18E (unknown)
SD04091 (no CG)
CG10641
SD04693
mesoderm
E
Late twist and Toll mesoderm genes
twi/wt Toll gof/wt
Prm (motor protein)
CG1942 (unknown)
CG16719 (unknown)
Ald (enzyme)
CG4945 (kinase)
CG3960 (actin binding)
CG18638 (enzyme)
CG5080 (myosin-like)
CG6378 (adhesion)
CG8154 (unknown)
Scg δ (CG14808)
CG5080
GH04349
somatic muscle
F
Mesoderm genes expressed very late in embryogenesis
twi/wt Toll gof/wt
CG9445 (unknown)
CG16882 (adhesion)
CG9509 (enzyme)
CG3987 (unknown)
CG6416 (LIM protein)
bt (projectin)
CG12896 (unknown)
CG5411 (enzyme)
Galpha73B (G-protein)
CG9480 (enzyme)
GH03983 (no CG)
CG16882
GH07717
muscle fibers

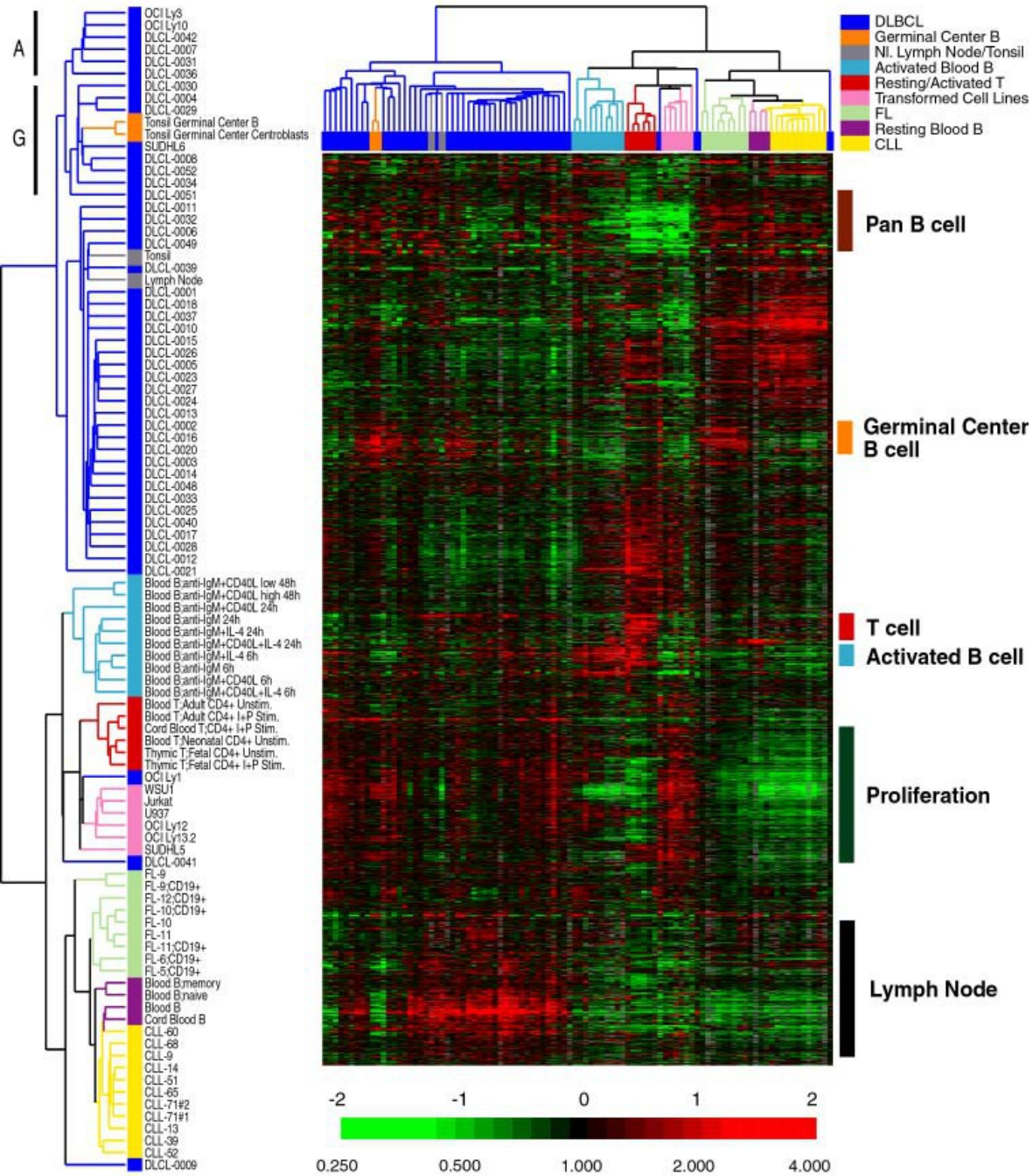
Example 3: clustering samples

A. Alizadeh *et al*., Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling, Nature 403: 503-11, 2000.

Response to treatment of patients w/ diffuse large B-cell lymphoma (DLBCL) is heterogeneous

Try to use expression data to discover finer distinctions among tumor types

Collected gene expression data for 42 DLBCL tumor samples + normal B-cells in various stages of differentiation + various controls


A
G
OCI Ly3
OCI Ly10
DLCL-0042
DLCL-0007
DLCL-0031
DLCL-0036
DLCL-0030
DLCL-0004
DLCL-0029
Tonsil Germinal Center B
Tonsil Germinal Center Centroblasts
SUDHL6
DLCL-0008
DLCL-0052
DLCL-0034
DLCL-0051
DLCL-0011
DLCL-0032
DLCL-0006
DLCL-0049
Tonsil
DLCL-0039
Lymph Node
DLCL-0001
DLCL-0018
DLCL-0037
DLCL-0010
DLCL-0015
DLCL-0026
DLCL-0005
DLCL-0023
DLCL-0027
DLCL-0024
DLCL-0013
DLCL-0002
DLCL-0016
DLCL-0020
DLCL-0003
DLCL-0014
DLCL-0048
DLCL-0033
DLCL-0025
DLCL-0040
DLCL-0017
DLCL-0028
DLCL-0012
DLCL-0021
Blood B;anti-IgM+CD40L low 48h
Blood B;anti-IgM+CD40L high 48h
Blood B;anti-IgM+CD40L 24h
Blood B;anti-IgM 24h
Blood B;anti-IgM+IL-4 24h
Blood B;anti-IgM+CD40L+IL-4 24h
Blood B;anti-IgM+IL-4 6h
Blood B;anti-IgM 6h
Blood B;anti-IgM+CD40L 6h
Blood B;anti-IgM+CD40L+IL-4 6h
Blood T;Adult CD4+ Unstim.
Blood T;Adult CD4+ I+P Stim.
Cord Blood T;CD4+ I+P Stim.
Blood T;Neonatal CD4+ Unstim.
Thymic T;Fetal CD4+ Unstim.
Thymic T;Fetal CD4+ I+P Stim.
OCI Ly1
WSU1
Jurkat
U937
OCI Ly12
OCI Ly13.2
SUDHL5
DLCL-0041
FL-9
FL-9;CD19+
FL-12;CD19+
FL-10;CD19+
FL-10
FL-11
FL-11;CD19+
FL-6;CD19+
FL-5;CD19+
Blood B;memory
Blood B;naive
Blood B
Cord Blood B
CLL-60
CLL-68
CLL-9
CLL-14
CLL-51
CLL-65
CLL-71#2
CLL-71#1
CLL-13
CLL-39
CLL-52
DLCL-0009
DLBCL
Germinal Center B
NI. Lymph Node/Tonsil
Activated Blood B
Resting/Activated T
Transformed Cell Lines
FL
Resting Blood B
CLL
Pan B cell
Germinal Center B cell
T cell
Activated B cell
Proliferation
Lymph Node
-2
-1
0
1
2
0.250
0.500
1.000
2.000
4.000

Everyday Usages

- Web search
- Image processing
- Data compression (Vector Quantization)
- Pattern recognition
- Data mining
- These days there are countless unstructured data streams which demand these types of algorithms
  - RFID tags
  - Web pages
  - Census data
  - Weather patterns
  - Anything you can think of!

K-means clustering

K-means is a partitional clustering algorithm

Let the set of data points (or instances) D be

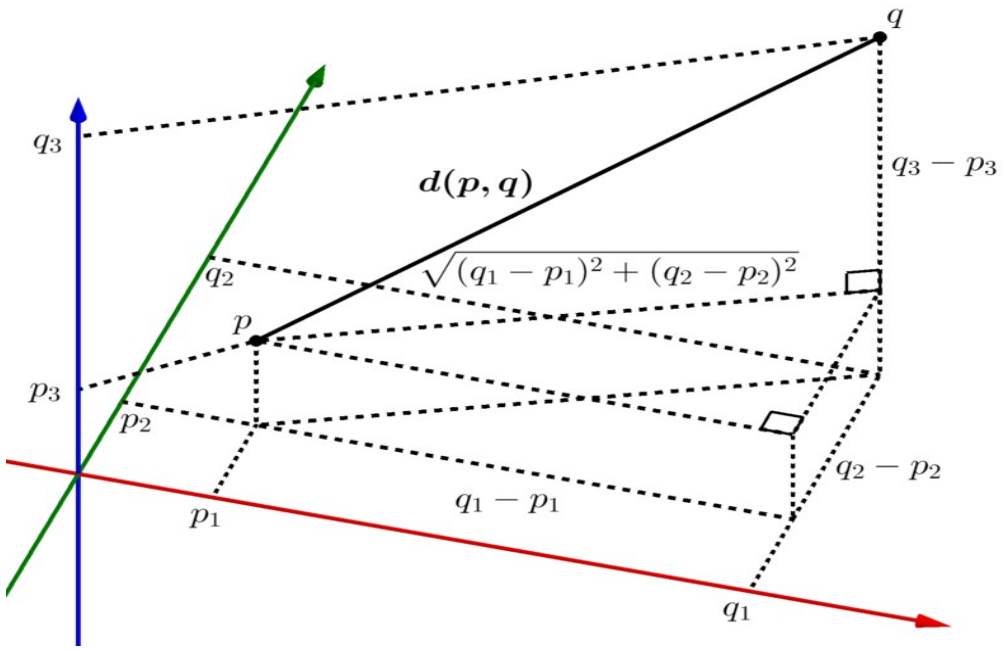
$$\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\},$$

where $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ir})$ is a vector in a real-valued space X ! R^r, and r is the number of attributes (dimensions) in the data.

The k-means algorithm partitions the given data into k clusters.

- Each cluster has a cluster **center**, called **centroid**.
- k is specified by the user

Measuring distance- Euclidean Distance



The **Euclidean distance** between points **p** and **q** is the length of the line segment connecting them ($\overline{\mathbf{pq}}$).

In Cartesian coordinates, if **p** = (p_1, p_2,..., p_n) and **q** = (q_1, q_2,..., q_n) are two points in Euclidean *n*-space, then the distance (d) from **p** to **q**, or from **q** to **p** is given by the Pythagorean formula:

$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1-p_1)^2+(q_2-p_2)^2+\cdots+(q_n-p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n}(q_i-p_i)^2}.$$

K-means algorithm

Given *k*, the *k-means* algorithm works as follows:

1) Randomly choose *k* data points (seeds) to be the initial centroids, cluster centers
2) Assign each data point to the closest centroid
3) Re-compute the centroids using the current cluster memberships.
4) If a convergence criterion is not met, go to 2).

K-means algorithm – (cont …)

Algorithm *k*-means(*k*, *D*)

1 Choose *k* data points as the initial centroids (cluster centers)
2 **repeat**
3 **for** each data point $\mathbf{x} \in D$ do
4 compute the distance from **x** to each centroid;
5 assign **x** to the closest centroid // a centroid represents a cluster
6 **endfor**
7 re-compute the centroids using the current cluster memberships
8 **until** the stopping criterion is met

Stopping/convergence criterion

1. no (or minimum) re-assignments of data points to different clusters,
2. no (or minimum) change of centroids, or
3. minimum decrease in the **sum of squared error** (SSE),

$$SSE = \sum_{j=1}^{k} \sum_{\mathbf{x} \in C_j} dist(\mathbf{x}, \mathbf{m}_j)^2$$

- C_i is the jth cluster, $\mathbf{m}_j$ is the centroid of cluster C_j (the mean vector of all the data points in C_j), and $dist(\mathbf{x}, \mathbf{m}_j)$ is the distance between data point $\mathbf{x}$ and centroid $\mathbf{m}_j$.

Sum of squared Errors

Error Sum of Squares (SSE)

SSE is the sum of the squared differences between each observation and its group's mean. It can be used as a measure of variation within a cluster. If all cases within a cluster are identical the SSE would then be equal to 0.

The formula for SSE is:

1.

$$SSE = \sum_{i=1}^{n} (x_i - \bar{x})^2$$

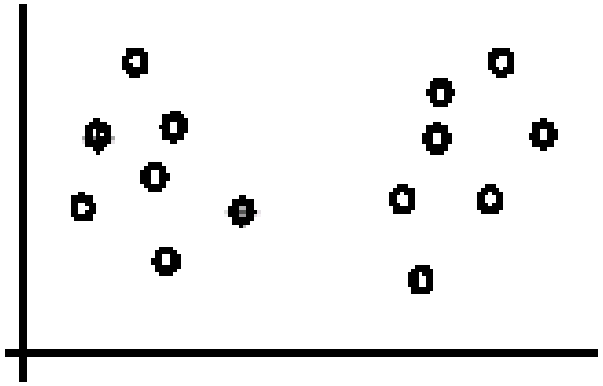
Where n is the number of observations x_i is the value of the ith observation and 0 is the mean of all the observations. This can also be rearranged to be written as seen in J.H. Ward's paper.

2.

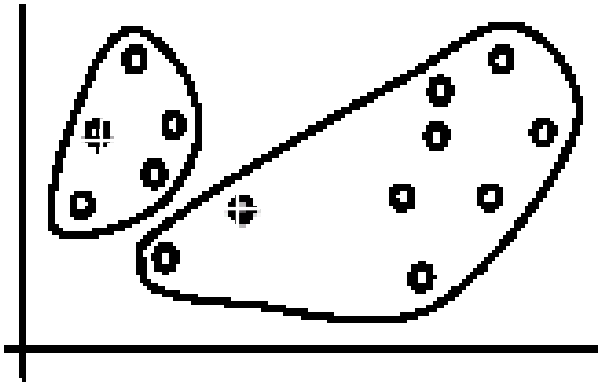
$$\mathrm{ESS} = \sum_{i=1}^{n} x_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)^2$$

At each stage of cluster analysis the total SSE is minimized with $SSE_{total} = SSE_1 + SSE_2 + SSE_3 + SSE_4 .... + SSE_n$. At the initial stage when each case is its own cluster this of course will be 0

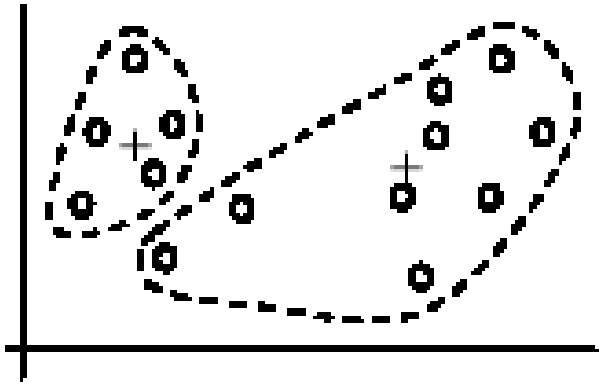
An example



(A). Random selection of k centers

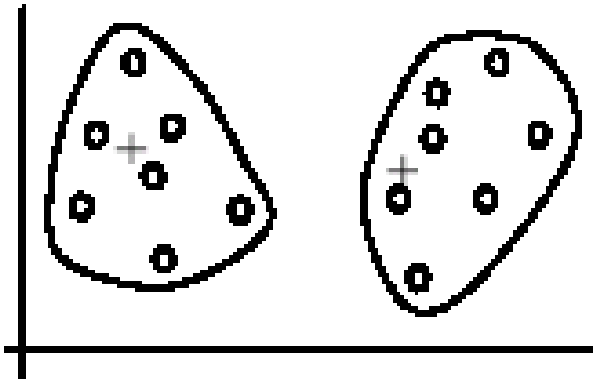


Iteration 1: (B). Cluster assignment

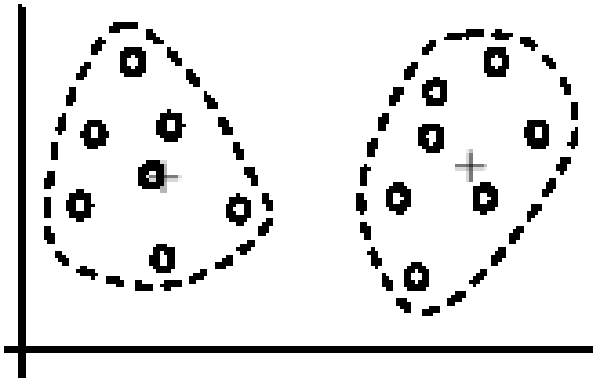


(C). Re-compute centroids

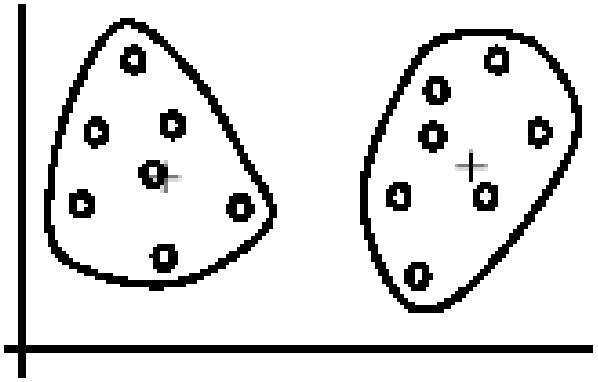
An example (cont …)



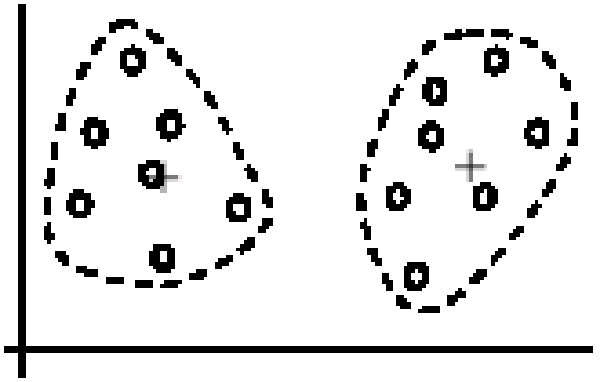
Iteration 2: (D). Cluster assignment



(E). Re-compute centroids



Iteration 3: (F). Cluster assignment



(G). Re-compute centroids

An example distance function

The *k*-means algorithm can be used for any application data set where the **mean** can be defined and computed. In the **Euclidean space**, the mean of a cluster is computed with:

$$\mathbf{m}_j = \frac{1}{|C_j|} \sum_{\mathbf{x}_i \in C_j} \mathbf{x}_i \tag{2}$$

where $|C_j|$ is the number of data points in cluster C_j. The distance from one data point $\mathbf{x}_i$ to a mean (centroid) $\mathbf{m}_j$ is computed with

$$dist(\mathbf{x}_i, \mathbf{m}_j) = \| \mathbf{x}_i - \mathbf{m}_j \| \tag{3}$$

$$= \sqrt{(x_{i1} - m_{j1})^2 + (x_{i2} - m_{j2})^2 + \ldots + (x_{ir} - m_{jr})^2}$$

A disk version of *k*-means

K-means can be implemented with data on disk

- In each iteration, it scans the data once.
- as the centroids can be computed incrementally

It can be used to cluster large datasets that do not fit in main memory

We need to control the number of iterations

- In practice, a limited is set (< 50).

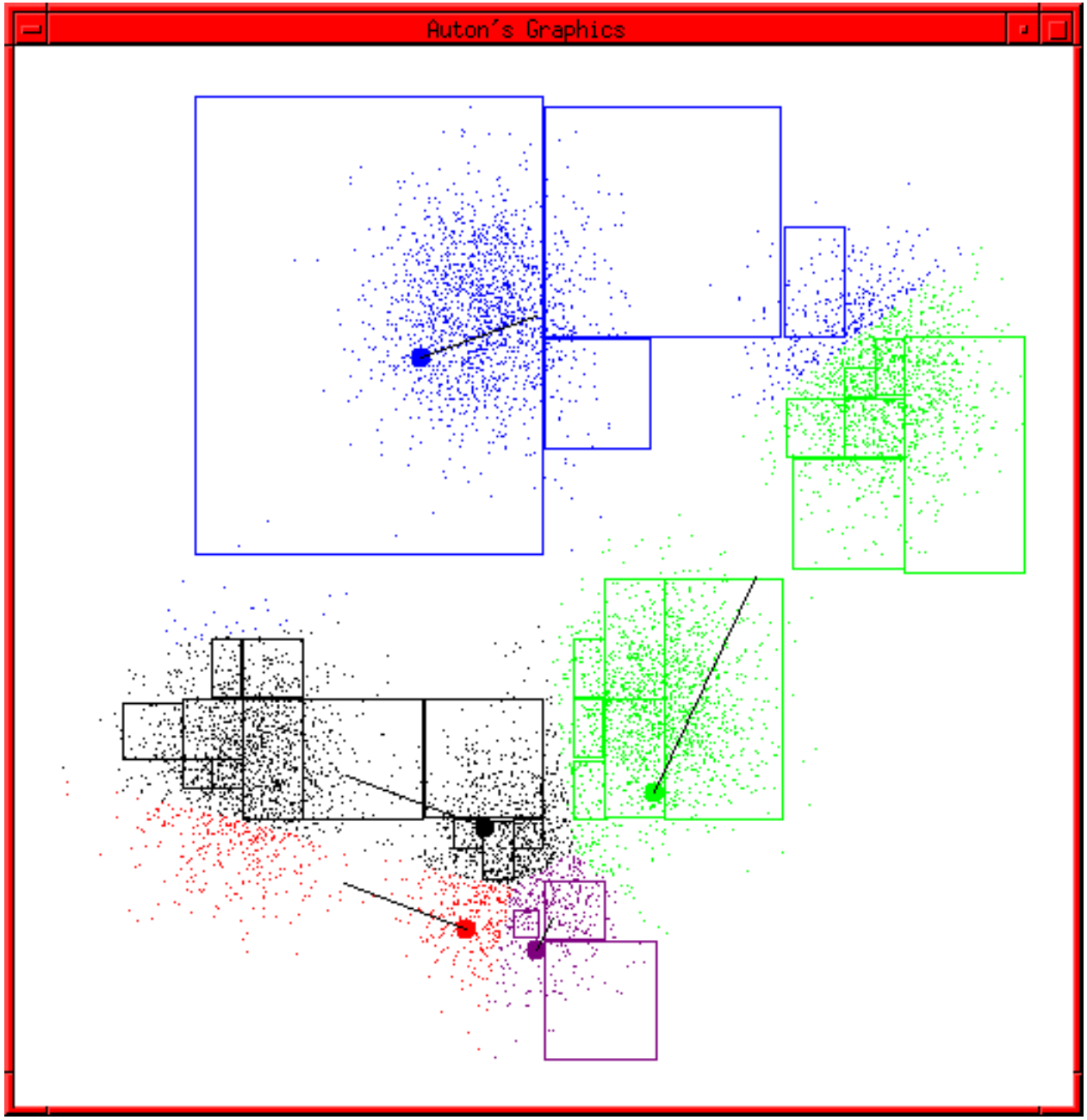
Not the best method. There are other scale-up algorithms, e.g., BIRCH.

A disk version of k-means (cont …)

Algorithm disk-k-means(k, D)

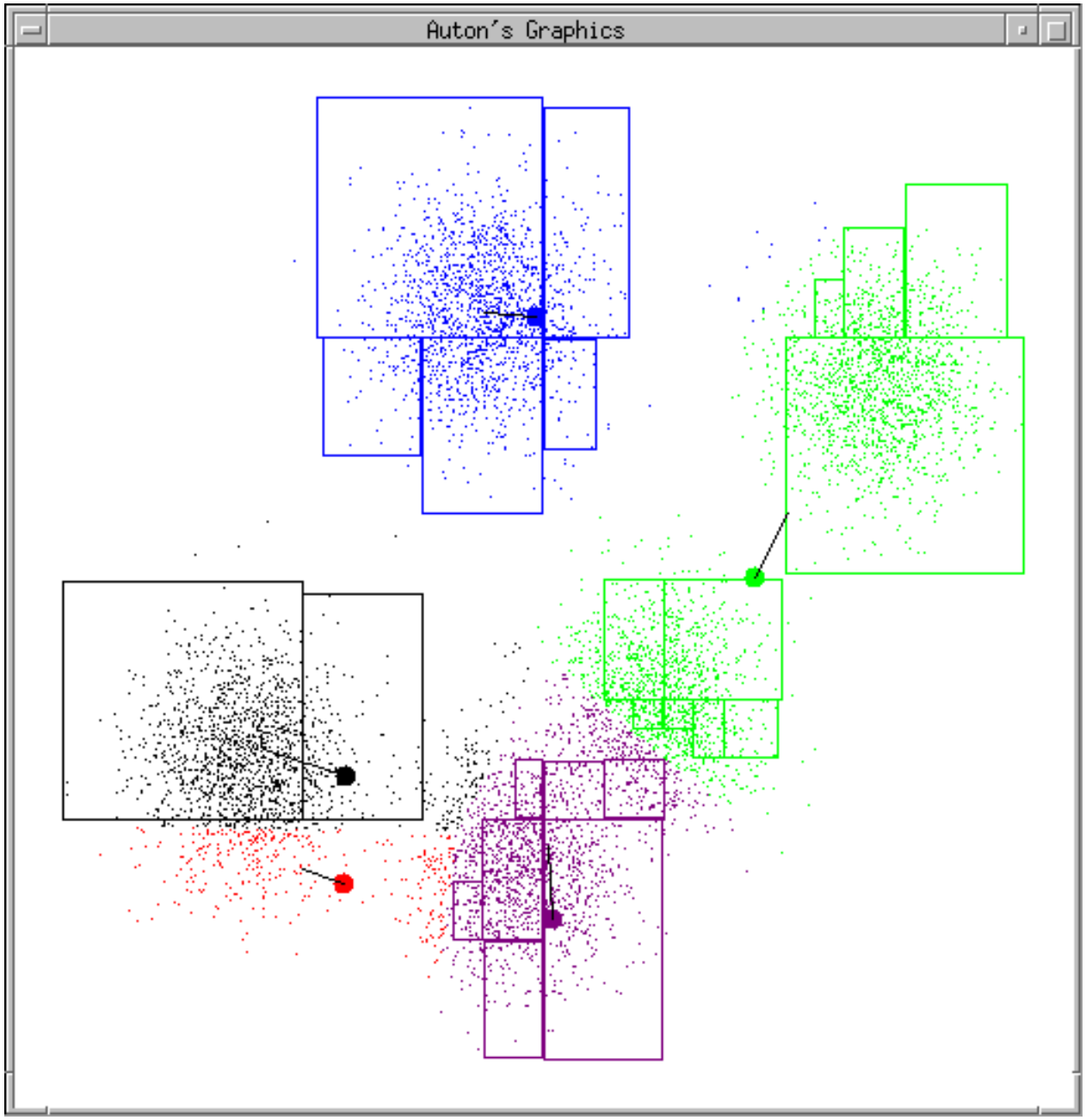
1 Choose k data points as the initial centriods $\mathbf{m}_j$, $j = 1, \ldots, k$;
2 **repeat**
3 initialize $\mathbf{s}_j = \mathbf{0}$, $j = 1, \ldots, k$; // $\mathbf{0}$ is a vector with all 0's
4 initialize $n_j = 0$, $j = 1, \ldots, k$; // n_j is the number points in cluster j
5 **for** each data point $\mathbf{x} \in D$ **do**
6 $j = \arg\min_j dist(\mathbf{x}, \mathbf{m}_j)$;
7 assign $\mathbf{x}$ to the cluster j;
8 $\mathbf{s}_j = \mathbf{s}_j + \mathbf{x}$;
9 $n_j = n_j + 1$;
10 **endfor**
11 $\mathbf{m}_i = \mathbf{s}_j / n_j$, $i = 1, \ldots, k$;
12 **until** the stopping criterion is met

K-means Start

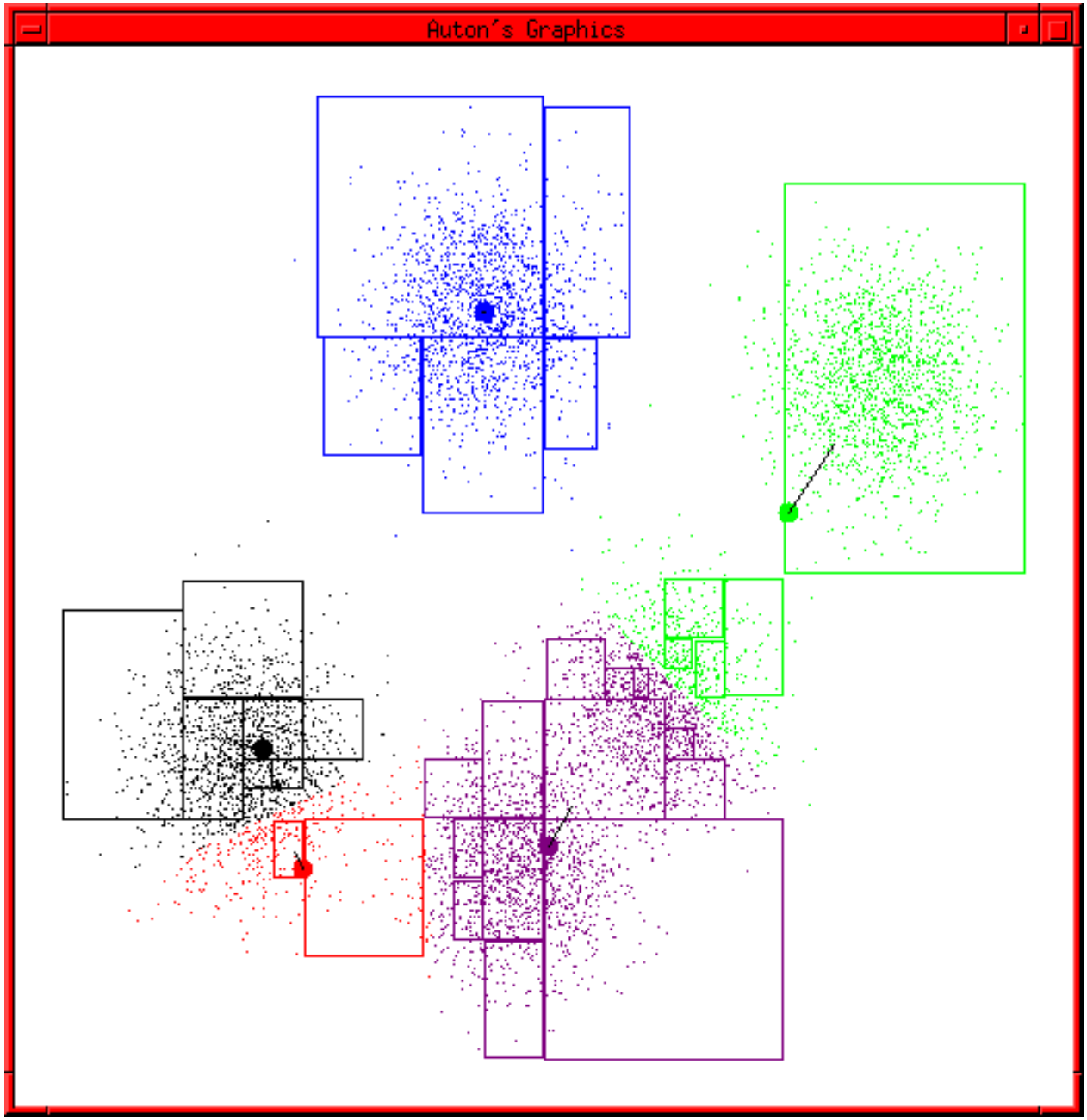


Dan Pelleg and Andrew Moore. Accelerating Exact k-means Algorithms with Geometric Reasoning. Proc. Conference on Knowledge Discovery in Databases 1999, (KDD99) (available on www.autonlab.org/pap.html*)*

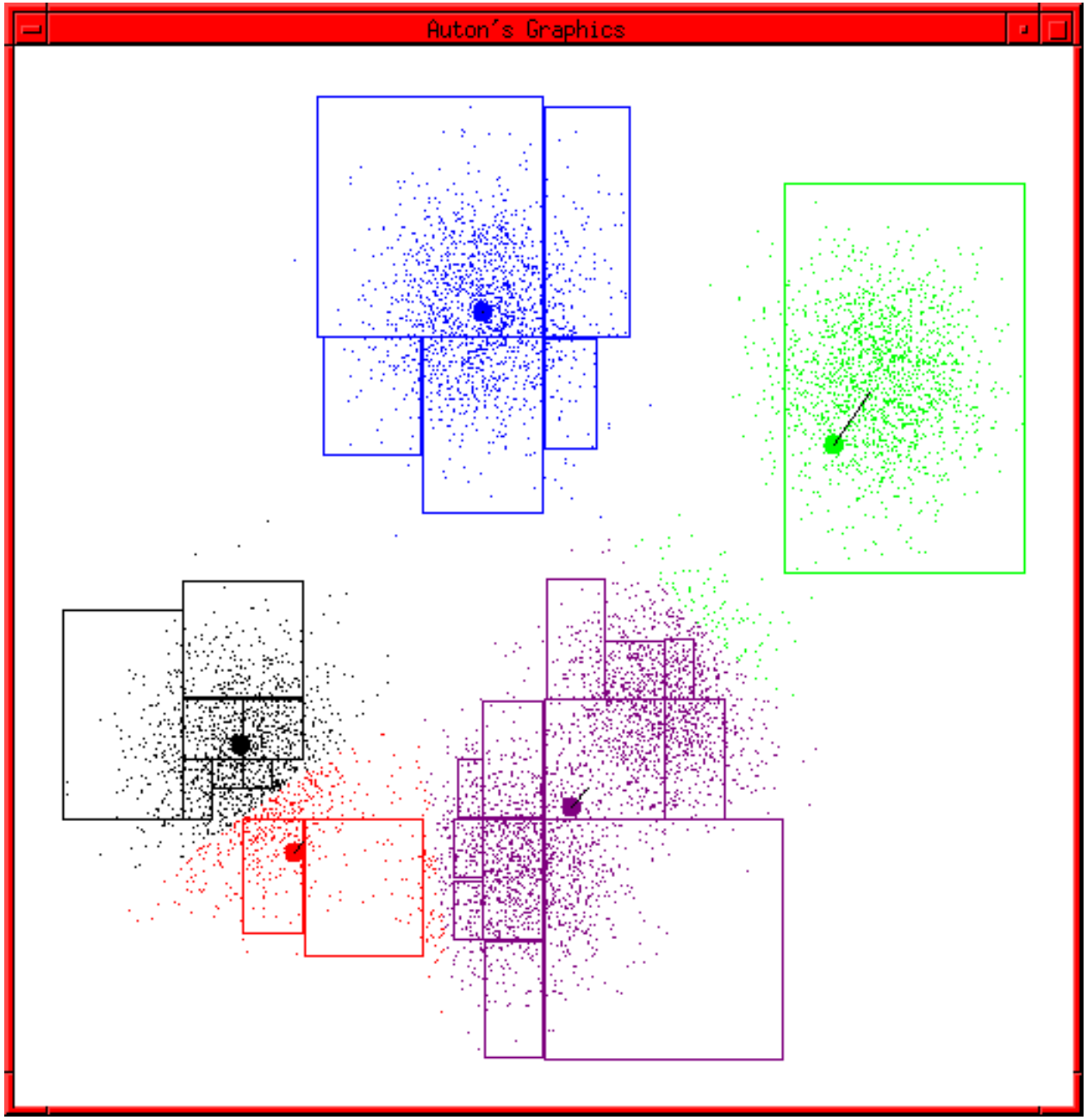
K-means continues…



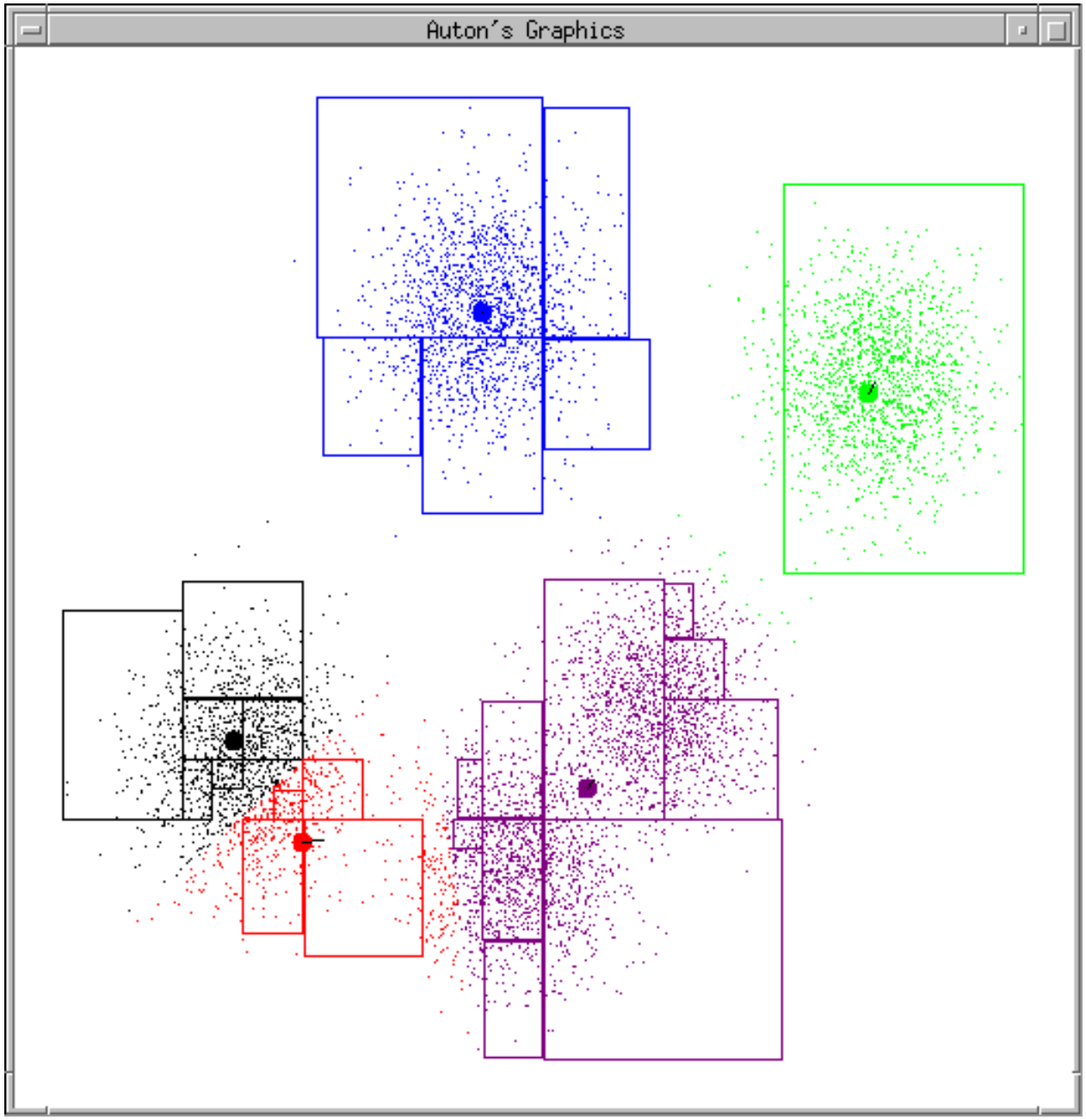
K-means continues…



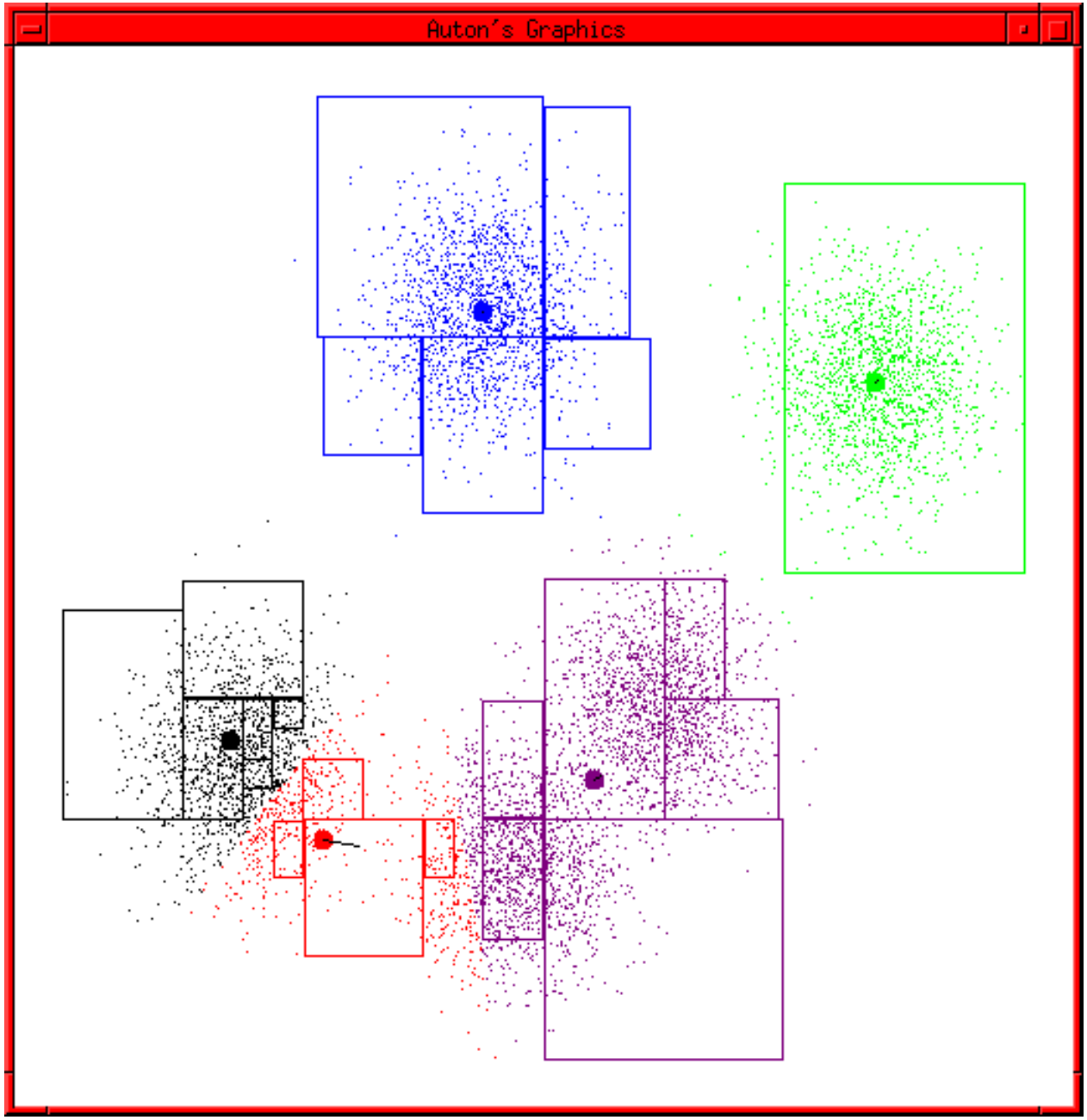
K-means continues…



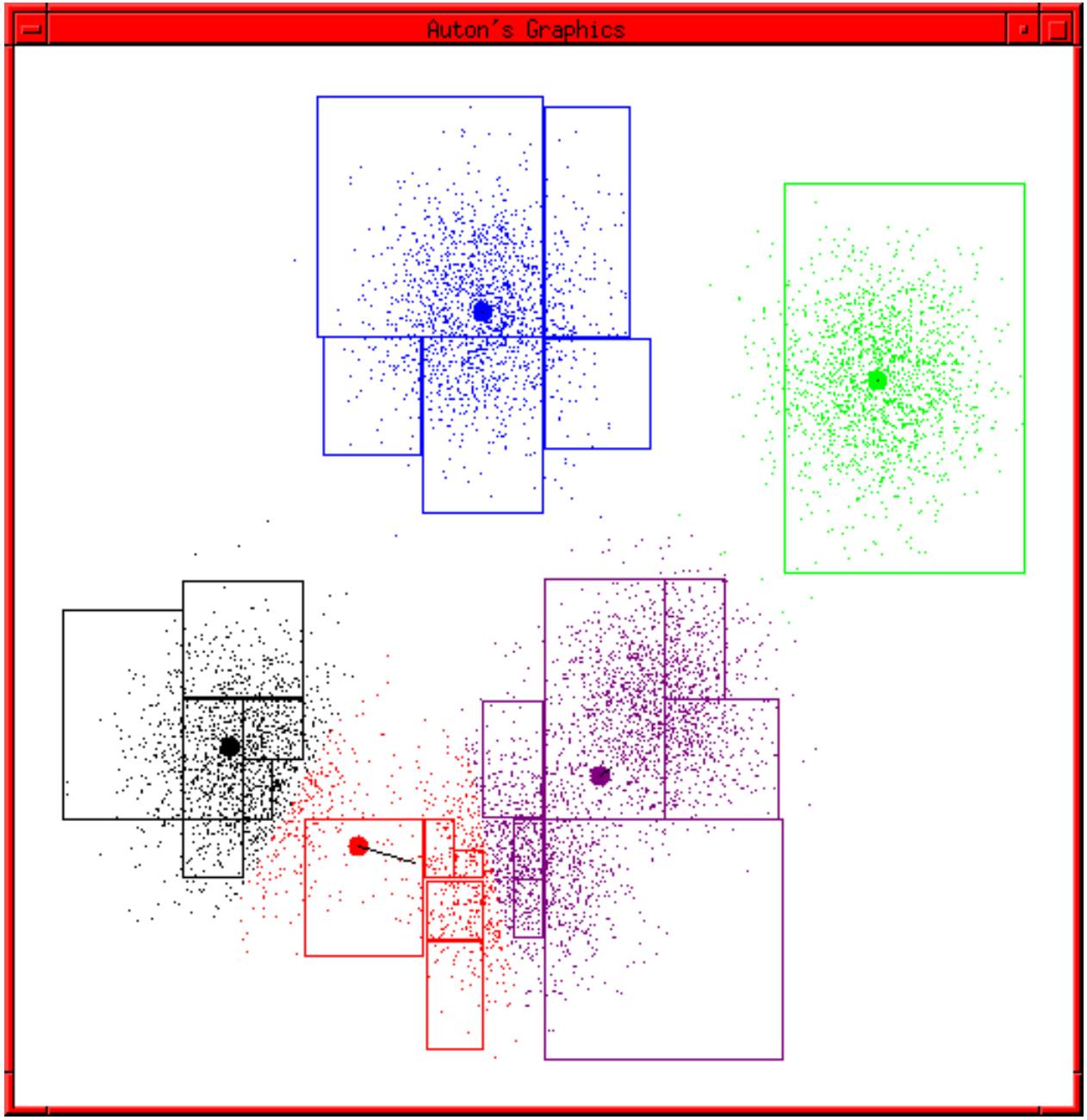
K-means continues…



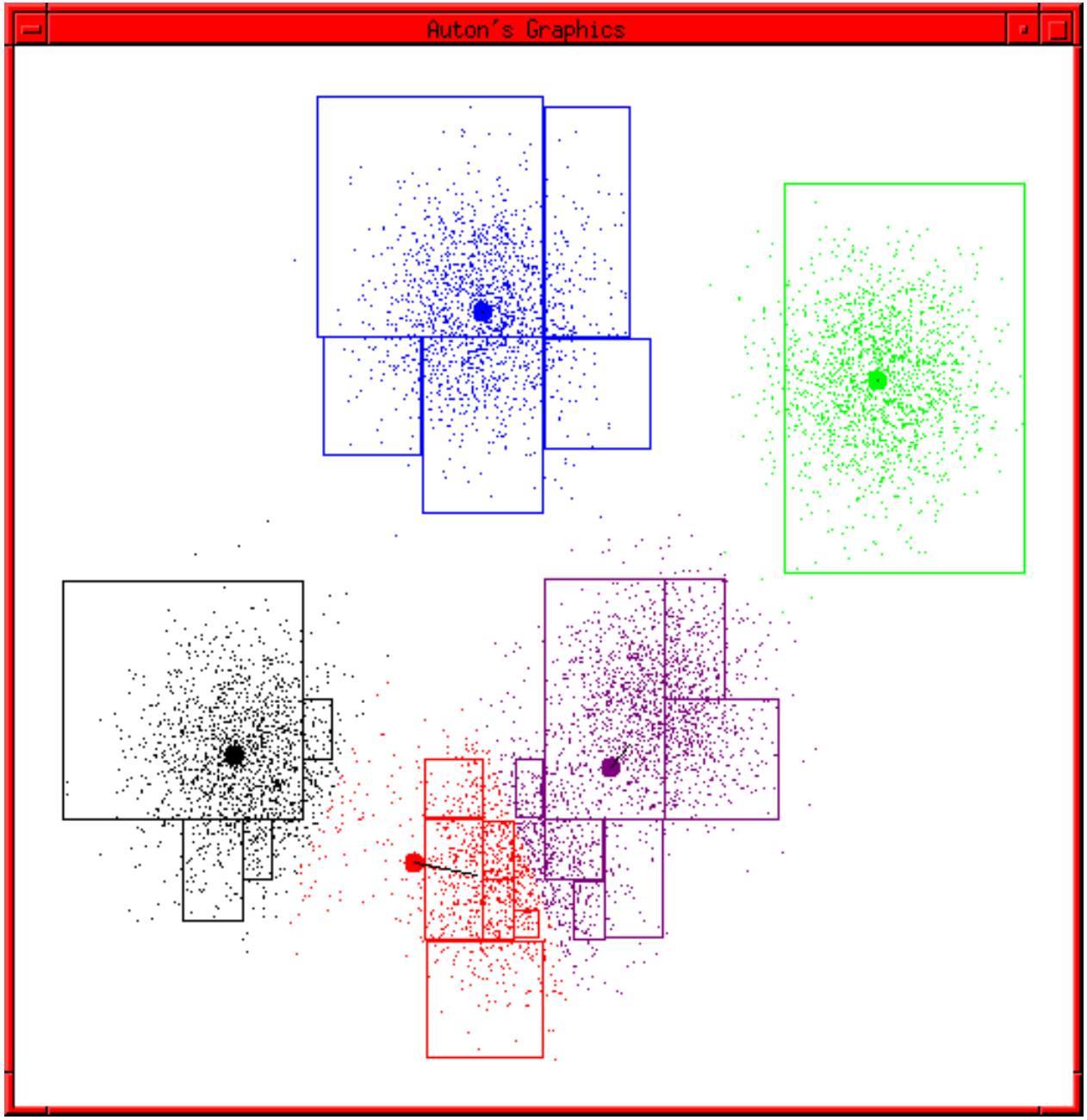
K-means continues…



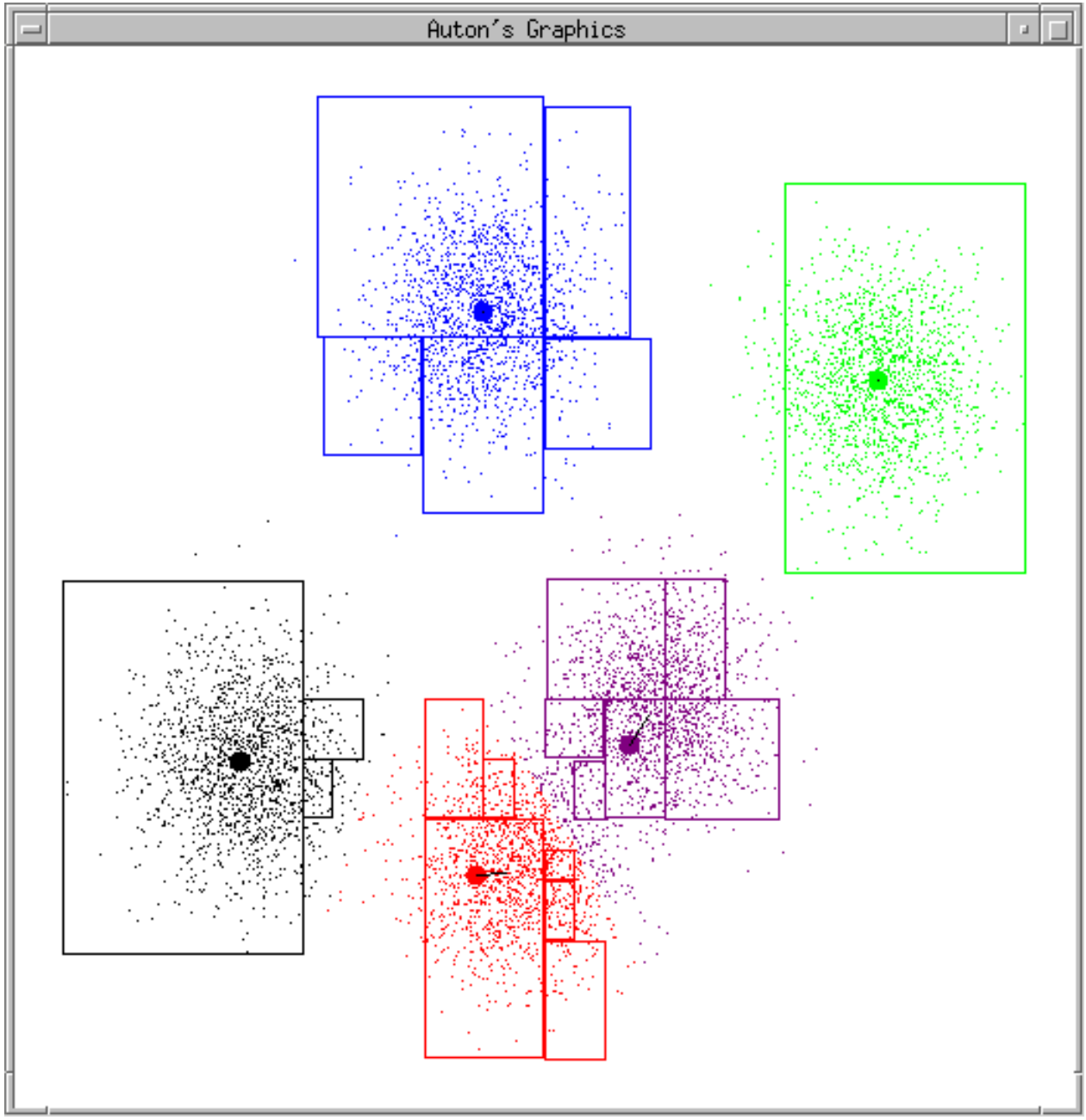
K-means continues…

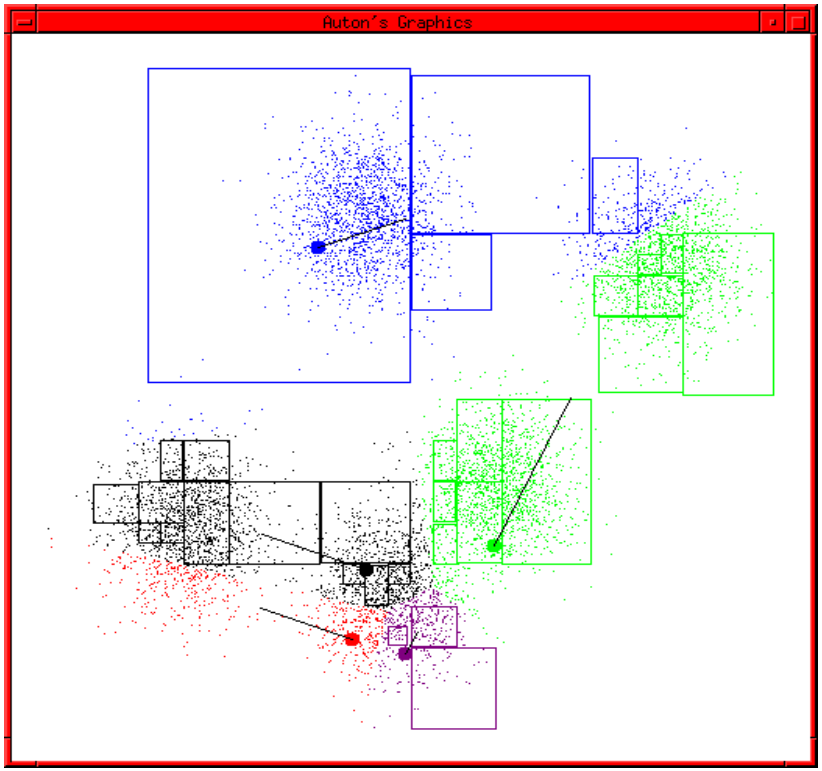


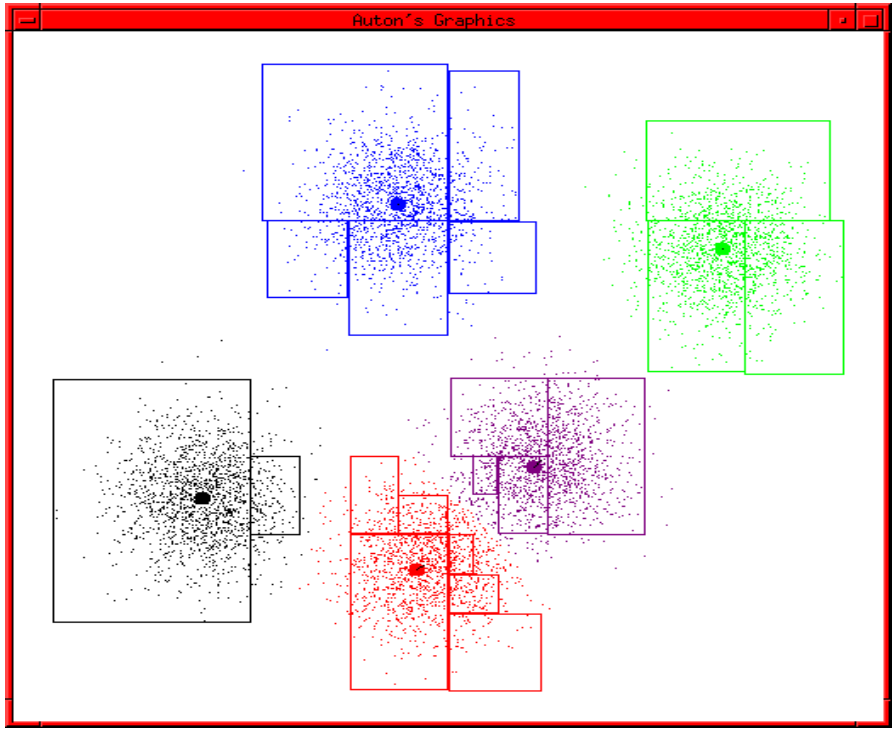
K-means continues…



K-means continues…




Auton's Graphics


Auton's Graphics

Weaknesses of k-means

The algorithm is only applicable if the mean is defined.

- For categorical data, *k*-mode - the centroid is represented by most frequent values.

The user needs to specify *k*.

The algorithm is sensitive to **outliers**

- Outliers are data points that are very far away from other data points.
- Outliers could be errors in the data recording or some special data points with very different values.

Strengths of k-means

Strengths:

- Simple: easy to understand and to implement
- Efficient: Time complexity: $O(tkn)$,

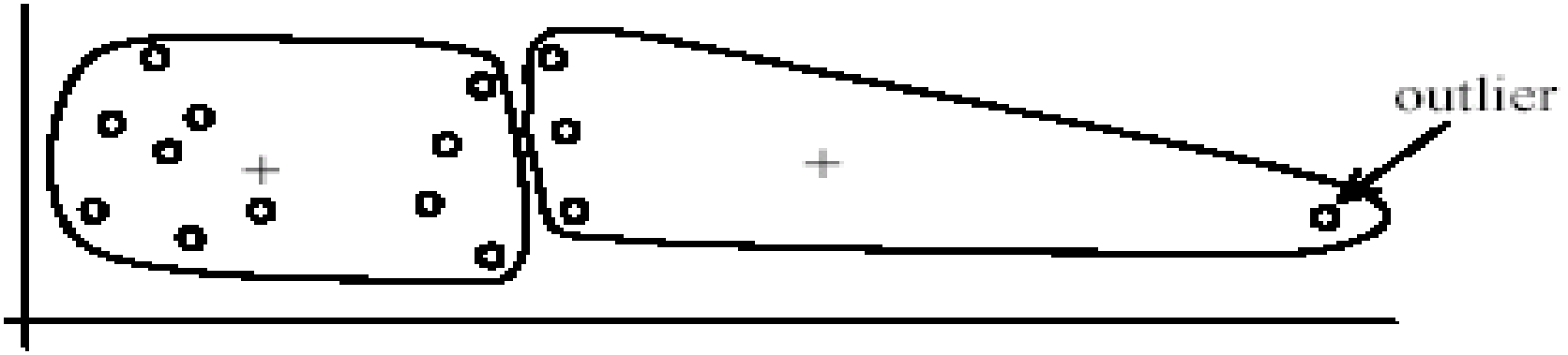
  where n is the number of data points,

  k is the number of clusters, and

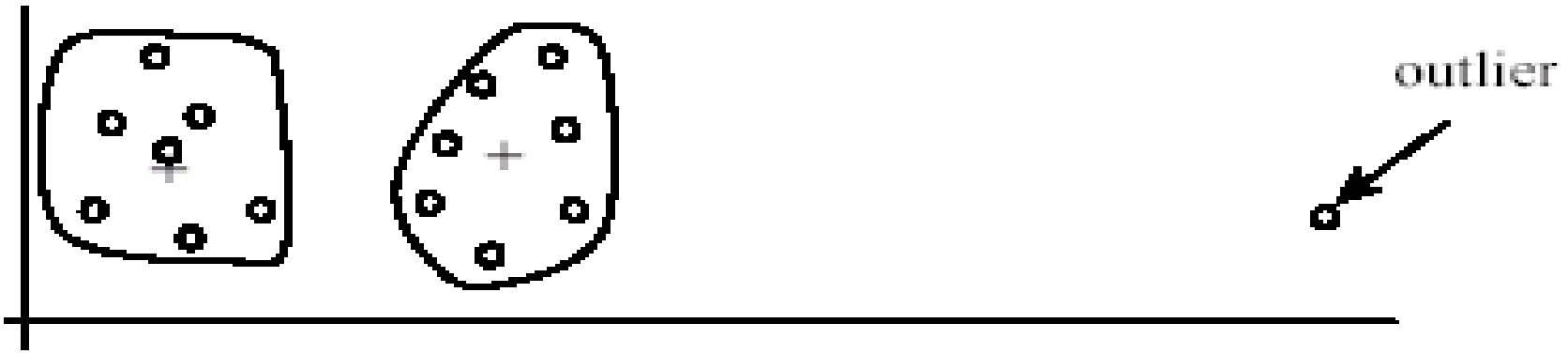
  t is the number of iterations.
- Since both k and t are small. k-means is considered a linear algorithm.

Note that: it terminates at a local optimum if SSE is used. The global optimum is hard to find due to complexity. :

Weaknesses of k-means: Problems with outliers



(A): Undesirable clusters



(B): Ideal clusters

Weaknesses of k-means: To deal with outliers

One method is to remove some data points in the clustering process that are much further away from the centroids than other data points.

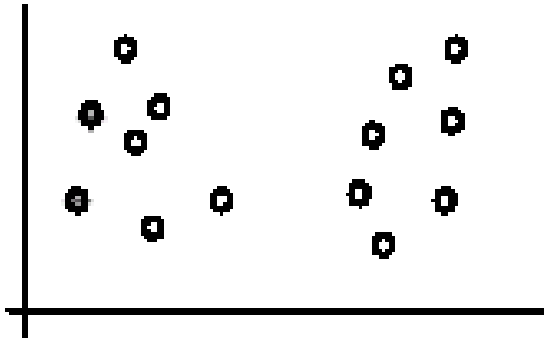
- To be safe, we may want to monitor these possible outliers over a few iterations and then decide to remove them.

Another method is to perform random sampling. Since in sampling we only choose a small subset of the data points, the chance of selecting an outlier is very small.

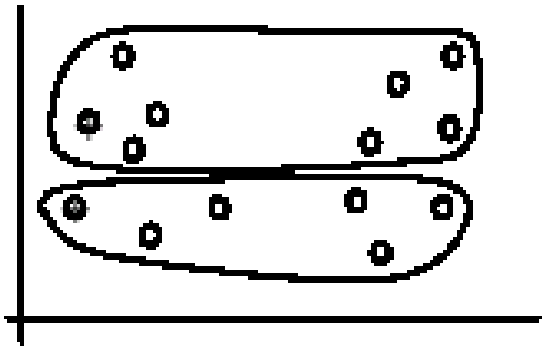
- Assign the rest of the data points to the clusters by distance or similarity comparison, or classification

Weaknesses of k-means (cont …)

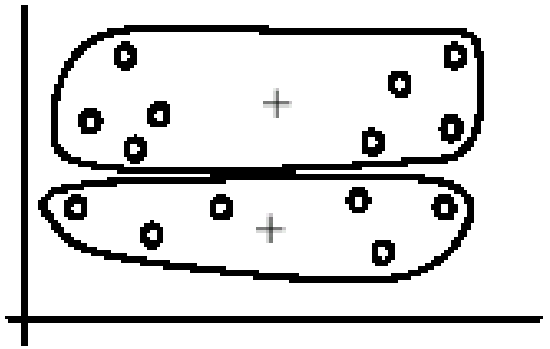
The algorithm is sensitive to initial seeds.



(A). Random selection of seeds (centroids)



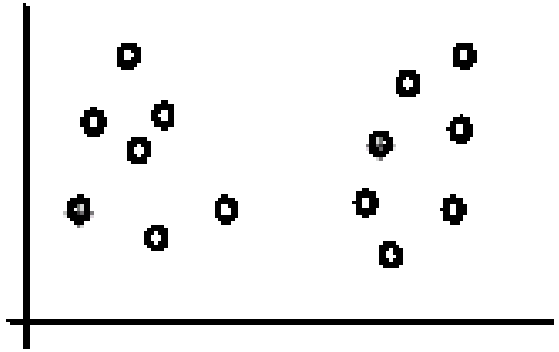
(B). Iteration 1



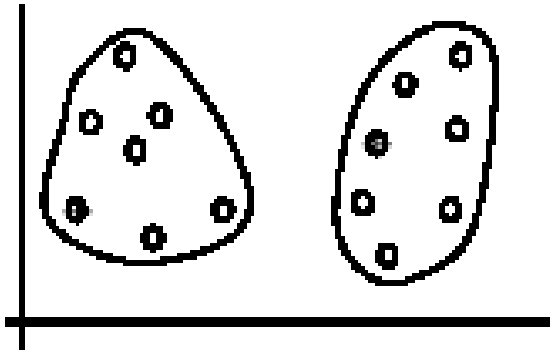
(C). Iteration 2

Weaknesses of k-means (cont …)

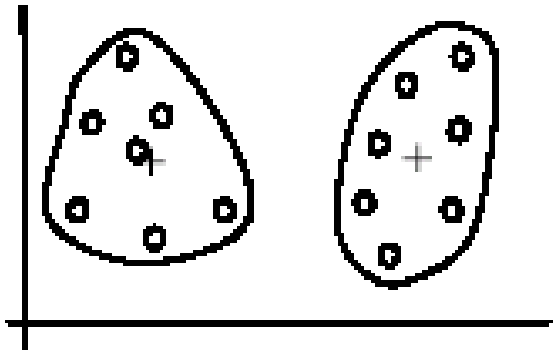
If we use different seeds: good results



(A). Random selection of *k* seeds (centroids)



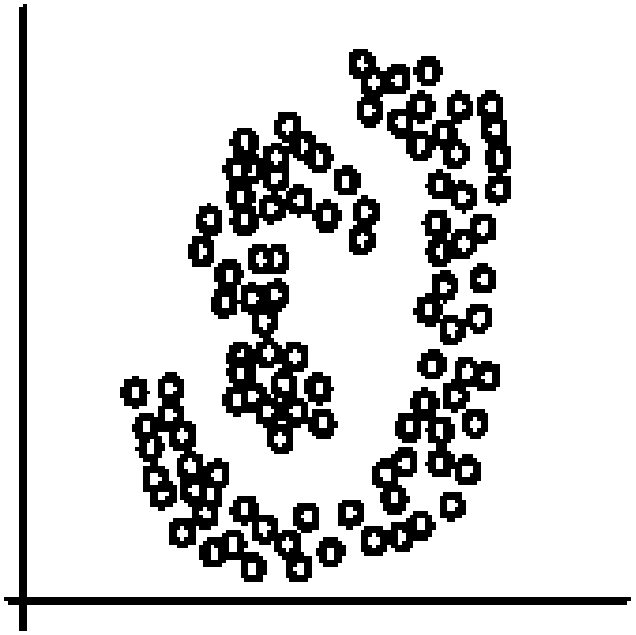
(B). Iteration 1



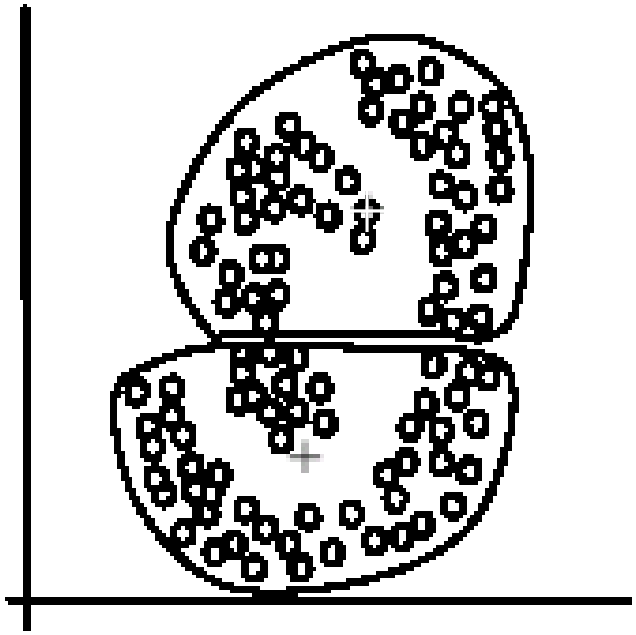
(C). Iteration 2

Weaknesses of k-means (cont …)

The *k*-means algorithm is not suitable for discovering clusters that are not hyper-ellipsoids (or hyper-spheres).



(A): Two natural clusters



(B): *k*-means clusters

K-means summary

Despite weaknesses, *k*-means is still the most popular algorithm due to its simplicity, efficiency and

- other clustering algorithms have their own lists of weaknesses.

No clear evidence that any other clustering algorithm performs better in general

- although they may be more suitable for some specific types of data or applications.

Comparing different clustering algorithms is a difficult task. No one knows the correct clusters!

Common ways to represent clusters

Use the centroid of each cluster to represent the cluster.

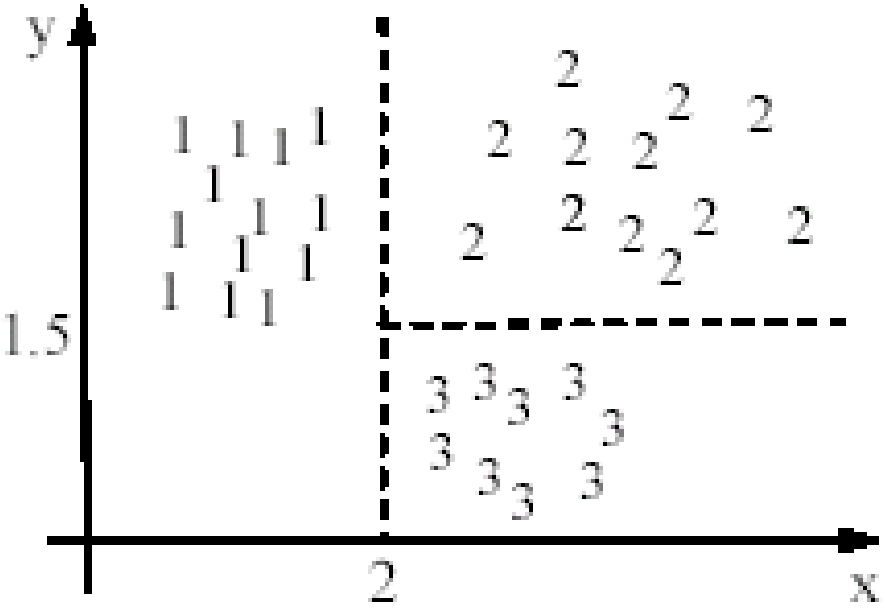
- compute the radius and
- standard deviation of the cluster to determine its spread in each dimension

- The centroid representation alone works well if the clusters are of the hyper-spherical shape.
- If clusters are elongated or are of other shapes, centroids are not sufficient

Using classification model

All the data points in a cluster are regarded to have the same class label, e.g., the cluster ID.

- run a supervised learning algorithm on the data to find a classification model.



$x \leq 2 \rightarrow$ cluster 1

$x > 2, y > 1.5 \rightarrow$ cluster 2

$x > 2, y \leq 1.5 \rightarrow$ cluster 3

Use frequent values to represent cluster

This method is mainly for clustering of categorical data (e.g., *k*-modes clustering).

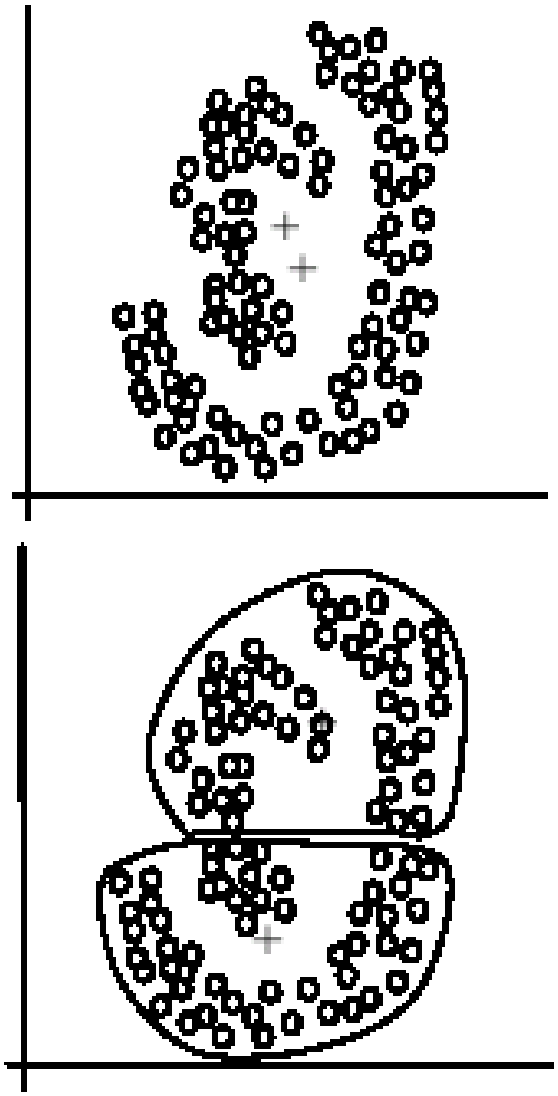
Main method used in text clustering, where a small set of frequent words in each cluster is selected to represent the cluster.

Clusters of arbitrary shapes

Hyper-elliptical and hyper-spherical clusters are usually easy to represent, using their centroid together with spreads.

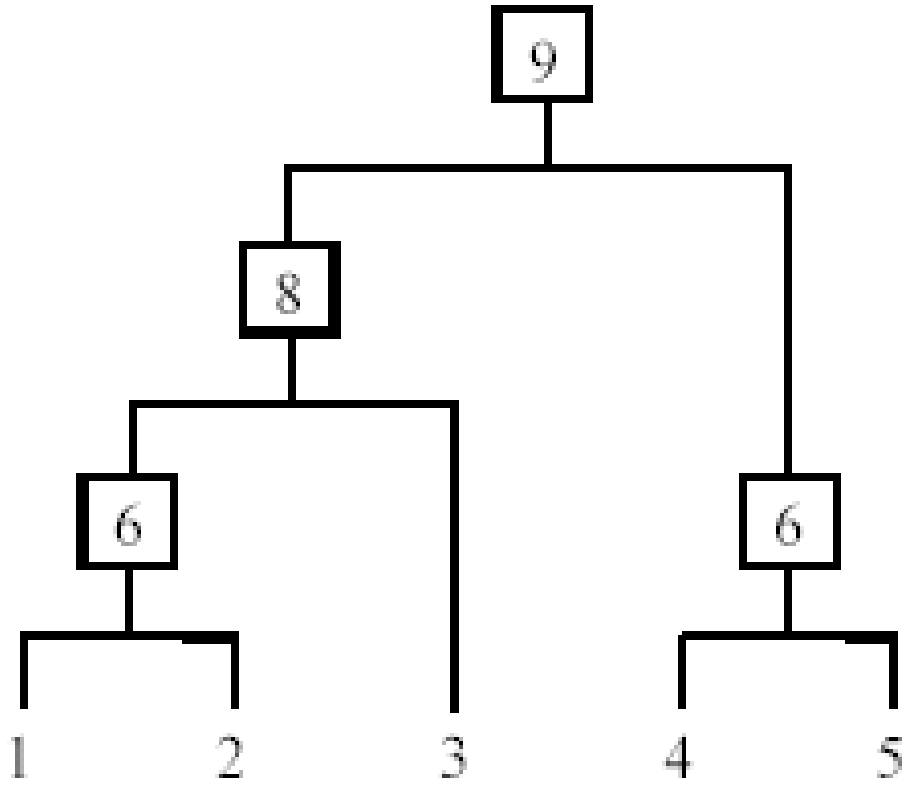
Irregular shape clusters are hard to represent. They may not be useful in some applications.

- Using centroids are not suitable (upper figure) in general
- K-means clusters may be more useful (lower figure), e.g., for making 2 size T-shirts.



Hierarchical Clustering

Produce a nested sequence of clusters, a tree, also called Dendrogram.



Types of hierarchical clustering

Agglomerative (bottom up) clustering: It builds the dendrogram (tree) from the bottom level, and

- merges the most similar (or nearest) pair of clusters
- stops when all the data points are merged into a single cluster (i.e., the root cluster).

Divisive (top down) clustering: It starts with all data points in one cluster, the root.

- Splits the root into a set of child clusters. Each child cluster is recursively divided further
- stops when only singleton clusters of individual data points remain, i.e., each cluster with only a single point

Agglomerative clustering

It is more popular then divisive methods.

At the beginning, each data point forms a cluster (also called a node).

Merge nodes/clusters that have the least distance.

Go on merging

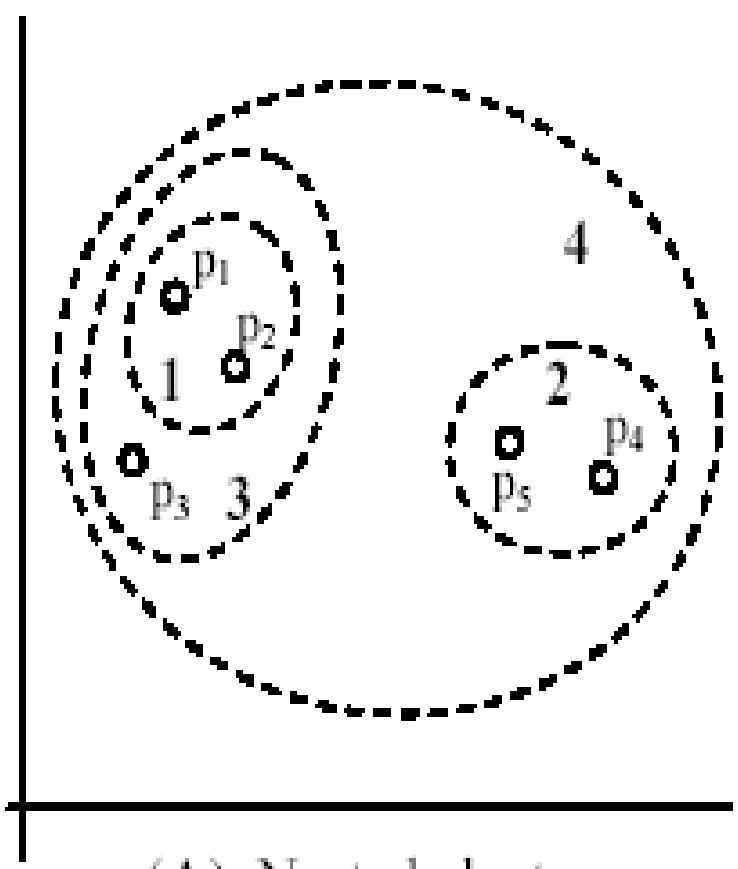
Eventually all nodes belong to one cluster

Agglomerative clustering algorithm

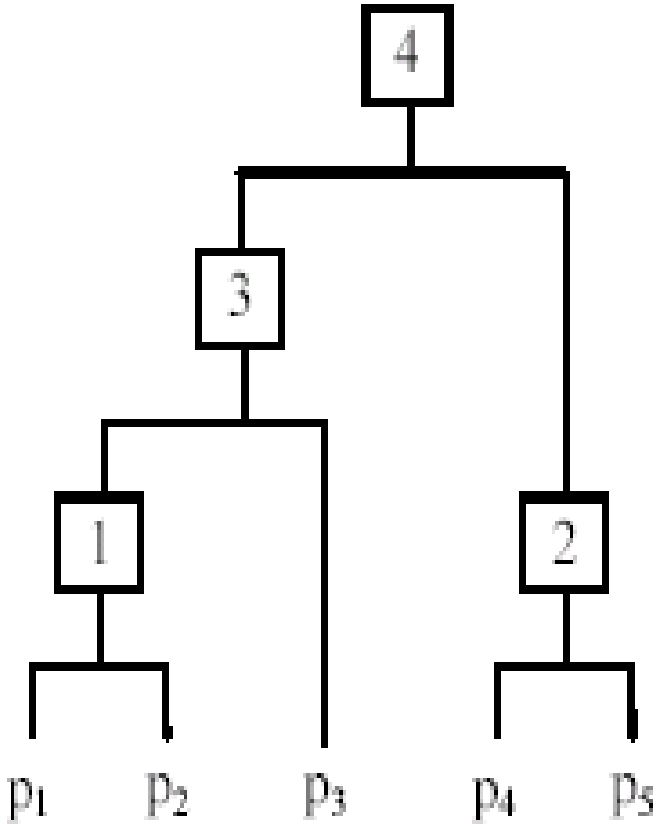
Algorithm Agglomerative(D)

1 Make each data point in the data set D a cluster,
2 Compute all pair-wise distances of $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in D$;
2 **repeat**
3 find two clusters that are nearest to each other;
4 merge the two clusters form a new cluster c;
5 compute the distance from c to all other clusters;
12 **until** there is only one cluster left

An example: working of the algorithm



(A). Nested clusters



(B) Dendrogram

Measuring the distance of two clusters

A few ways to measure distances of two clusters.

Results in different variations of the algorithm.

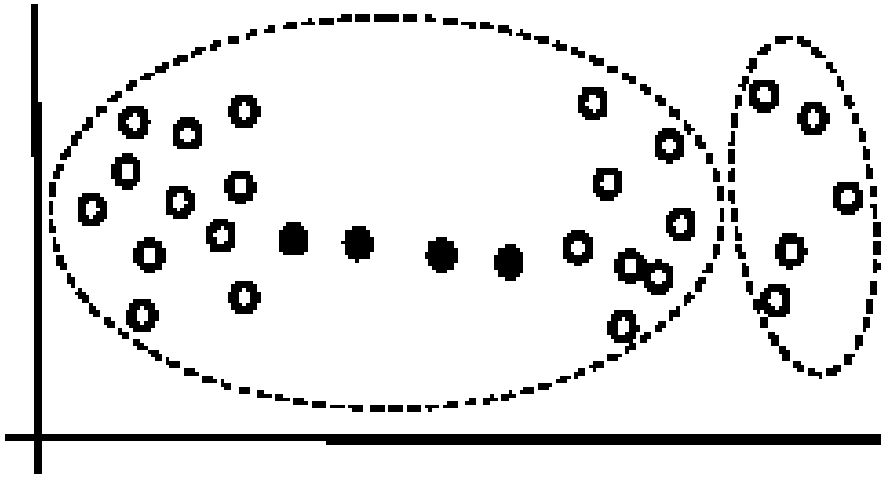
- Single link
- Complete link
- Average link
- Centroids
- …

Single link method

The distance between two clusters is the distance between two closest data points in the two clusters, one data point from each cluster.

It can find arbitrarily shaped clusters, but

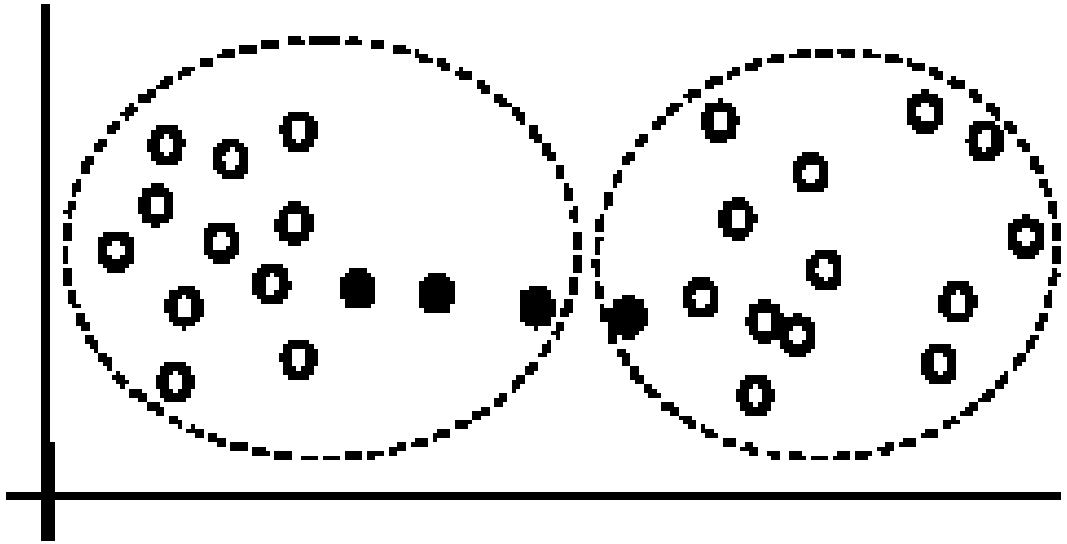
- It may cause the undesirable "chain effect" by noisy points



Complete link method

The distance between two clusters is the distance of two furthest data points in the two clusters.

It is sensitive to outliers because they are far away



Average link and centroid methods

Average link: A compromise between

- the sensitivity of complete-link clustering to outliers and
- the tendency of single-link clustering to form long chains that do not correspond to the intuitive notion of clusters as compact, spherical objects.
- In this method, the distance between two clusters is the average distance of all pair-wise distances between the data points in two clusters.

Centroid method: In this method, the distance between two clusters is the distance between their centroids

The complexity

All the algorithms are at least $O(n^2)$. n is the number of data points.

Single link can be done in $O(n^2)$.

Complete and average links can be done in $O(n^2 \log n)$.

Due the complexity, hard to use for large data sets.

- Sampling
- Scale-up methods (e.g., BIRCH).

Which clustering algorithm to use?

How to choose a clustering algorithm

Clustering research has a long history. A vast collection of algorithms are available.

- We only introduced several main algorithms.

Choosing the "best" algorithm is a challenge.

- Every algorithm has limitations and works well with certain data distributions.
- It is very hard, if not impossible, to know what distribution the application data follow. The data may not fully follow any "ideal" structure or distribution required by the algorithms.
- One also needs to decide how to standardize the data, to choose a suitable distance function and to select other parameter values.

Choose a clustering algorithm (cont …)

Due to these complexities, the common practice is to

- run several algorithms using different distance functions and parameter settings, and
- then carefully analyze and compare the results.

The interpretation of the results must be based on insight into the meaning of the original data together with knowledge of the algorithms used.

Clustering is highly application dependent and to certain extent subjective (personal preferences).

Cluster Evaluation: hard problem

The quality of a clustering is very hard to evaluate because

- We do not know the correct clusters

Some methods are used:

- User inspection
  - Study centroids, and spreads
  - Rules from a decision tree.
  - For text documents, one can read some documents in clusters.

Cluster evaluation: ground truth

We use some labeled data (for classification)

Assumption: Each class is a cluster.

After clustering, a confusion matrix is constructed. From the matrix, we compute various measurements, entropy, purity, precision, recall and F-score.

- Let the classes in the data D be $C = (c_1, c_2, \ldots, c_k)$. The clustering method produces k clusters, which divides D into k disjoint subsets, D_1, D_2, …, D_k.

Evaluation measures: Entropy

Entropy: For each cluster, we can measure its entropy as follows:

$$entropy(D_i) = -\sum_{j=1}^{k} \Pr_i(c_j) \log_2 \Pr_i(c_j), \qquad (29)$$

where $\Pr_i(c_j)$ is the proportion of class c_j data points in cluster i or D_i. The total entropy of the whole clustering (which considers all clusters) is

$$entropy_{total}(D) = \sum_{i=1}^{k} \frac{|D_i|}{|D|} \times entropy(D_i) \qquad (30)$$

A remark about ground truth evaluation

Commonly used to compare different clustering algorithms.

A real-life data set for clustering has no class labels.

- Thus although an algorithm may perform very well on some labeled data sets, no guarantee that it will perform well on the actual application data at hand.

The fact that it performs well on some label data sets does give us some confidence of the quality of the algorithm.

This evaluation method is said to be based on **external data** or information.

Evaluation based on internal information

Intra-cluster cohesion (compactness):

- Cohesion measures how near the data points in a cluster are to the cluster centroid.
- Sum of squared error (SSE) is a commonly used measure.

I**nter-cluster separation** (isolation):

- Separation means that different cluster centroids should be far away from one another.

In most applications, expert judgments are still the key.

Indirect evaluation

In some applications, clustering is not the primary task, but used to help perform another task.

We can use the performance on the primary task to compare clustering methods.

For instance, in an application, the primary task is to provide recommendations on book purchasing to online shoppers.

- If we can cluster books according to their features, we might be able to provide better recommendations.
- We can evaluate different clustering algorithms based on how well they help with the recommendation task.
- Here, we assume that the recommendation can be reliably evaluated.

Indirect evaluation

In some applications, clustering is not the primary task, but used to help perform another task.

We can use the performance on the primary task to compare clustering methods.

For instance, in an application, the primary task is to provide recommendations on book purchasing to online shoppers.

- If we can cluster books according to their features, we might be able to provide better recommendations.
- We can evaluate different clustering algorithms based on how well they help with the recommendation task.
- Here, we assume that the recommendation can be reliably evaluated.

Holes in data space

- All the clustering algorithms only group data.
- Clusters only represent one aspect of the knowledge in the data.
- Another aspect that we have not studied is the **holes**.
  - A hole is a region in the data space that contains no or few data points. Reasons:
    - insufficient data in certain areas, and/or
    - certain attribute-value combinations are not possible or seldom occur.

Holes are useful too

Although clusters are important, holes in the space can be quite useful too.

For example, in a disease database

- we may find that certain symptoms and/or test values do not occur together, or
- when a certain medicine is used, some test values never go beyond certain ranges.

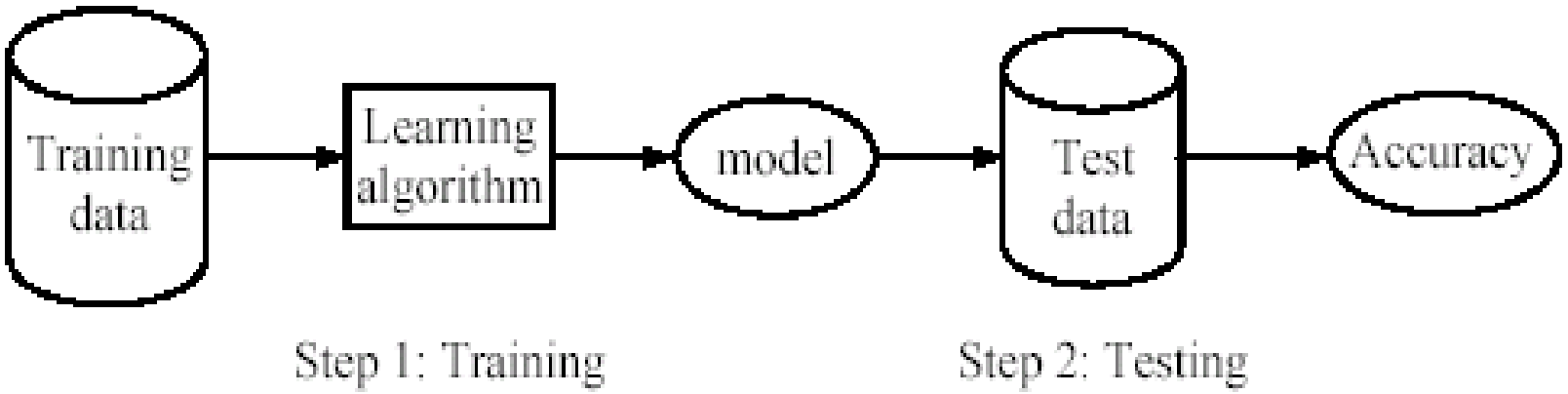
Discovery of such information can be important in medical domains because

- it could mean the discovery of a cure to a disease or some biological laws.

Supervised learning process: two steps

- Learning (training)
  training data
- Testing: unseen test data

$$Accuracy\ !\ \frac{\text{Number of correct classifications}}{\text{Total number of test cases}},$$



Avoid overfitting in classification

Overfitting: A tree may overfit the training data

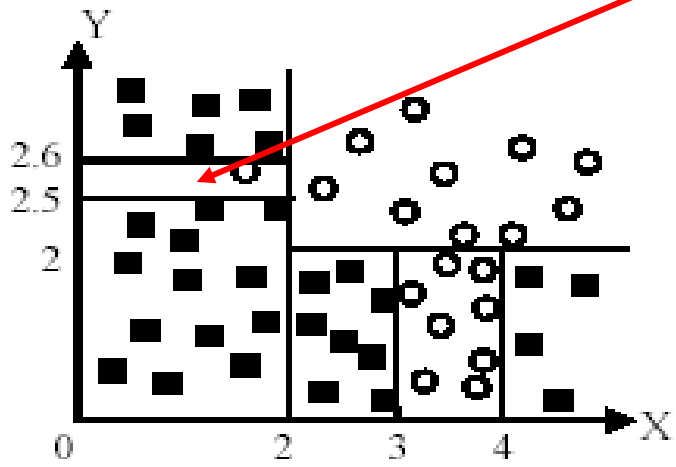
- Good accuracy on training data but poor on test data
- Symptoms: tree too deep and too many branches, some may reflect anomalies due to noise or outliers

Two approaches to avoid overfitting

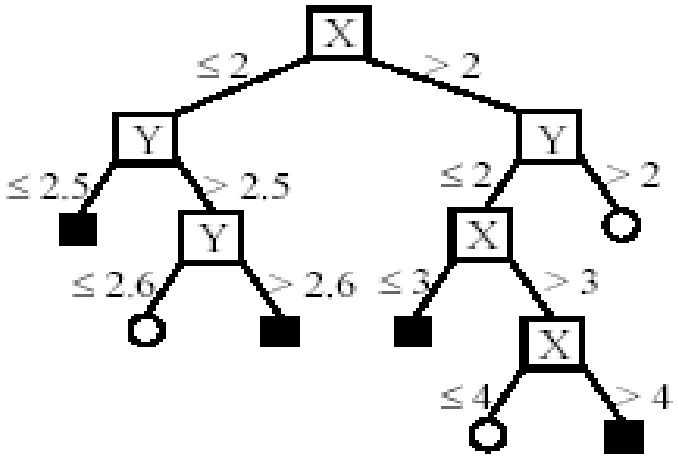
- Pre-pruning: Halt tree construction early
  - Difficult to decide because we do not know what may happen subsequently if we keep growing the tree.
- Post-pruning: Remove branches or sub-trees from a "fully grown" tree.
  - This method is commonly used. C4.5 uses a statistical method to estimates the errors at each node for pruning.
  - A validation set may be used for pruning as well.

An example

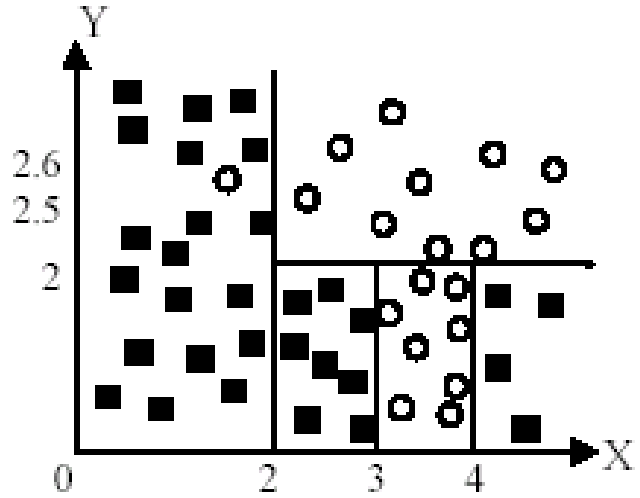
Likely to overfit the data

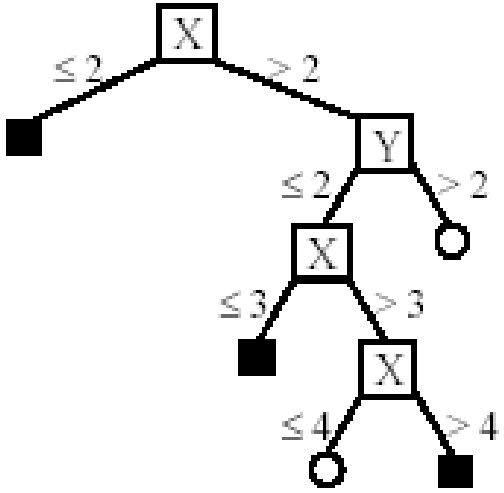


(A) A partition of the data space



(B). The decision tree





Other issues in decision tree learning

From tree to rules, and rule pruning

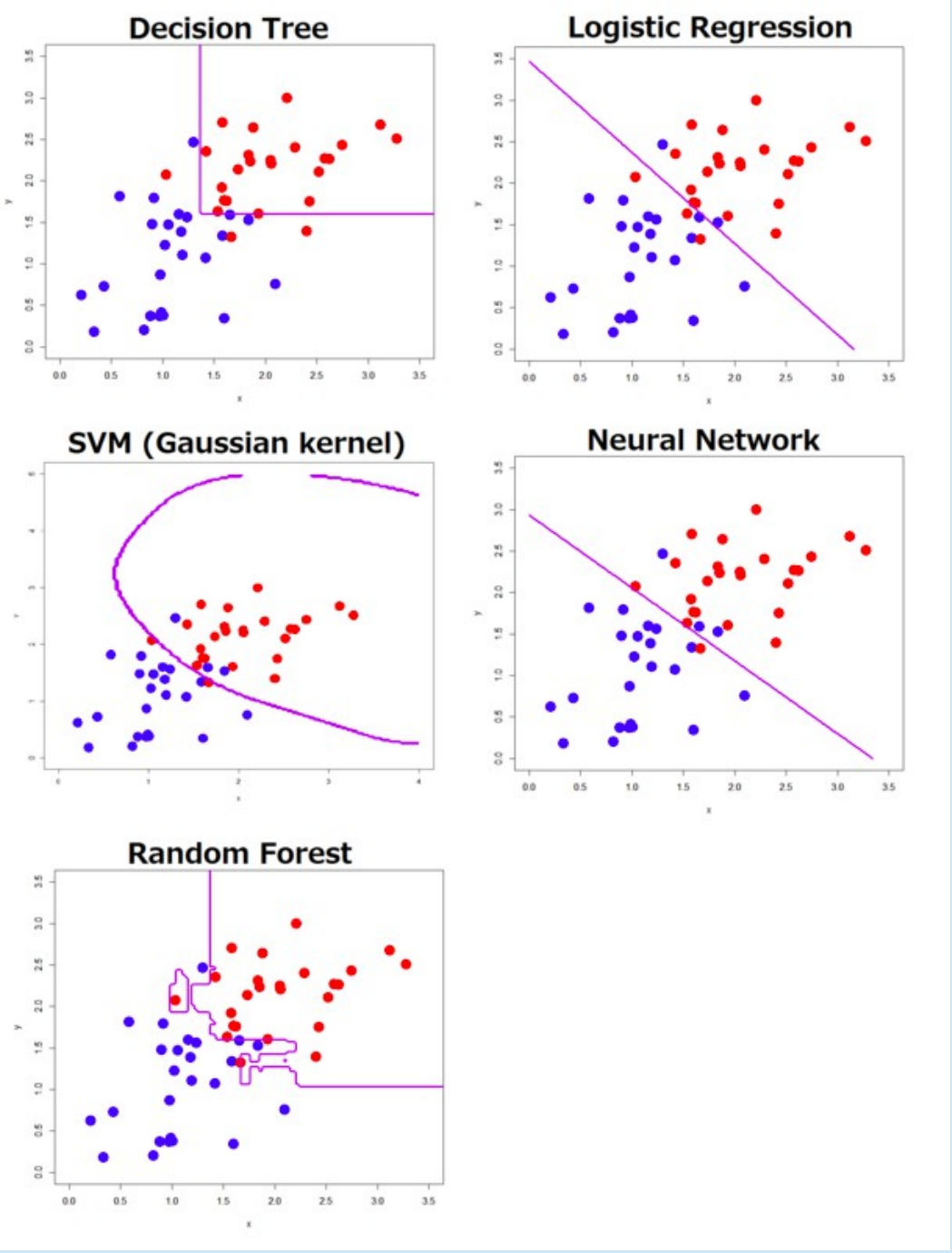
Handling of miss values

Handing skewed distributions

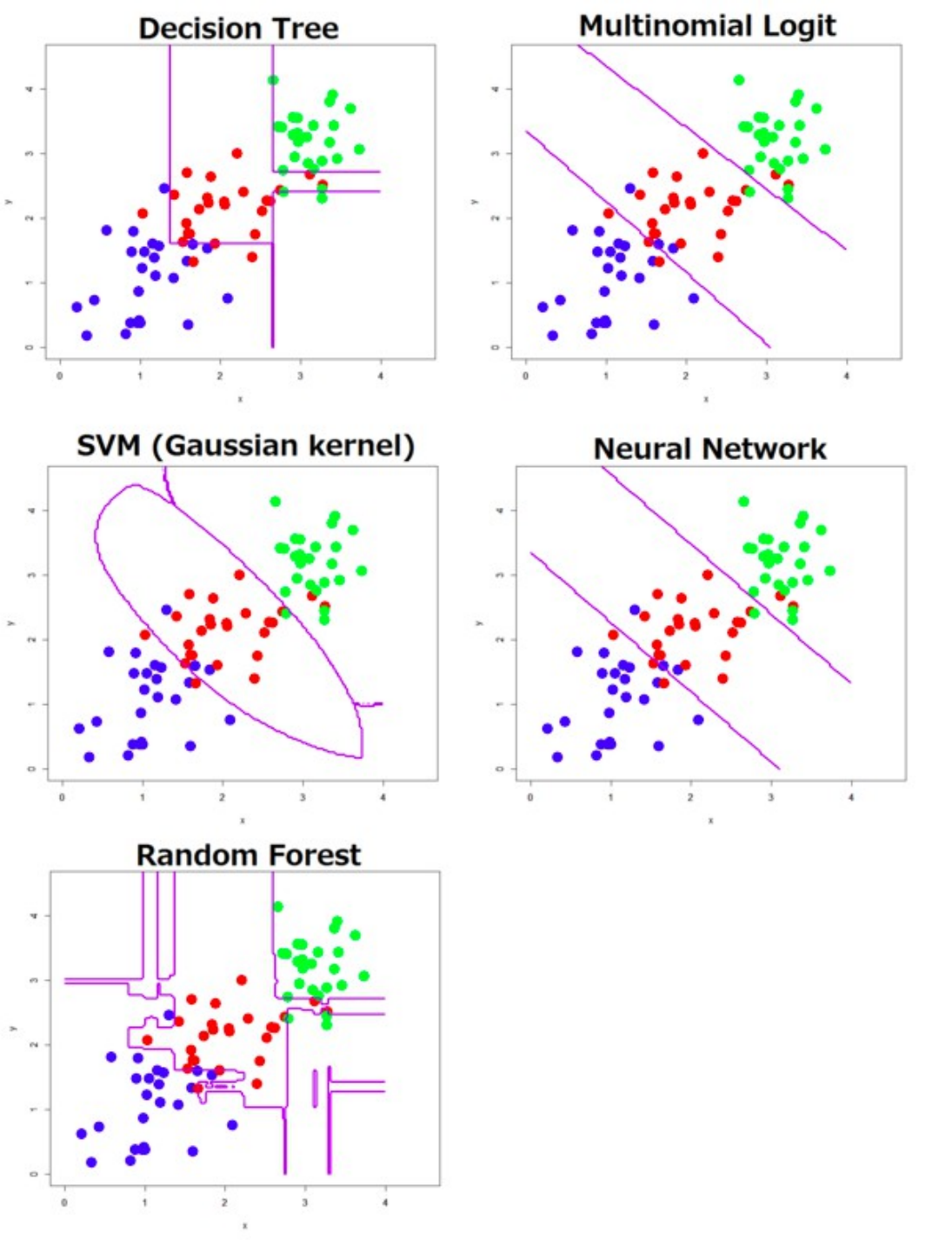
Handling attributes and classes with different costs.

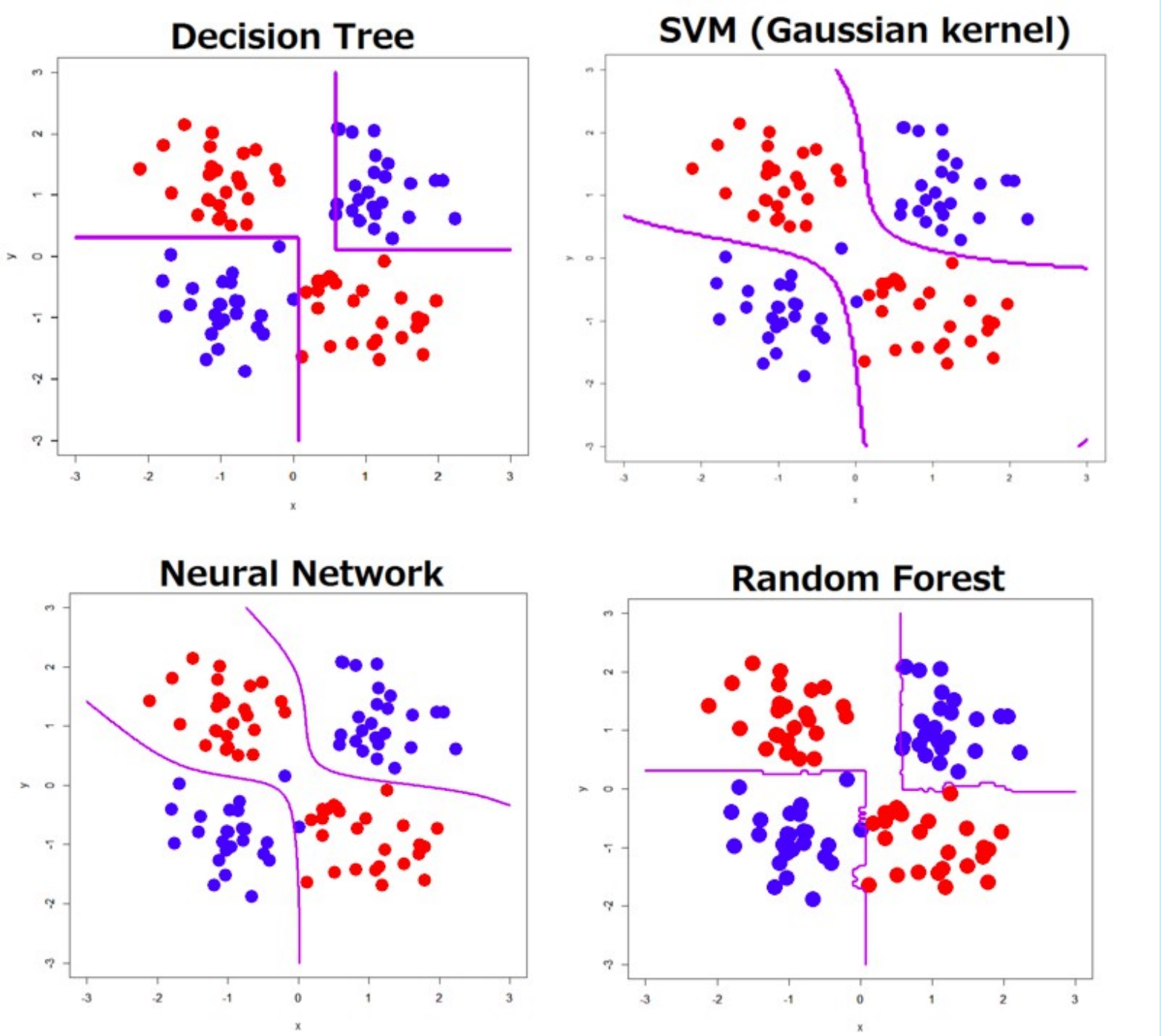
Attribute construction

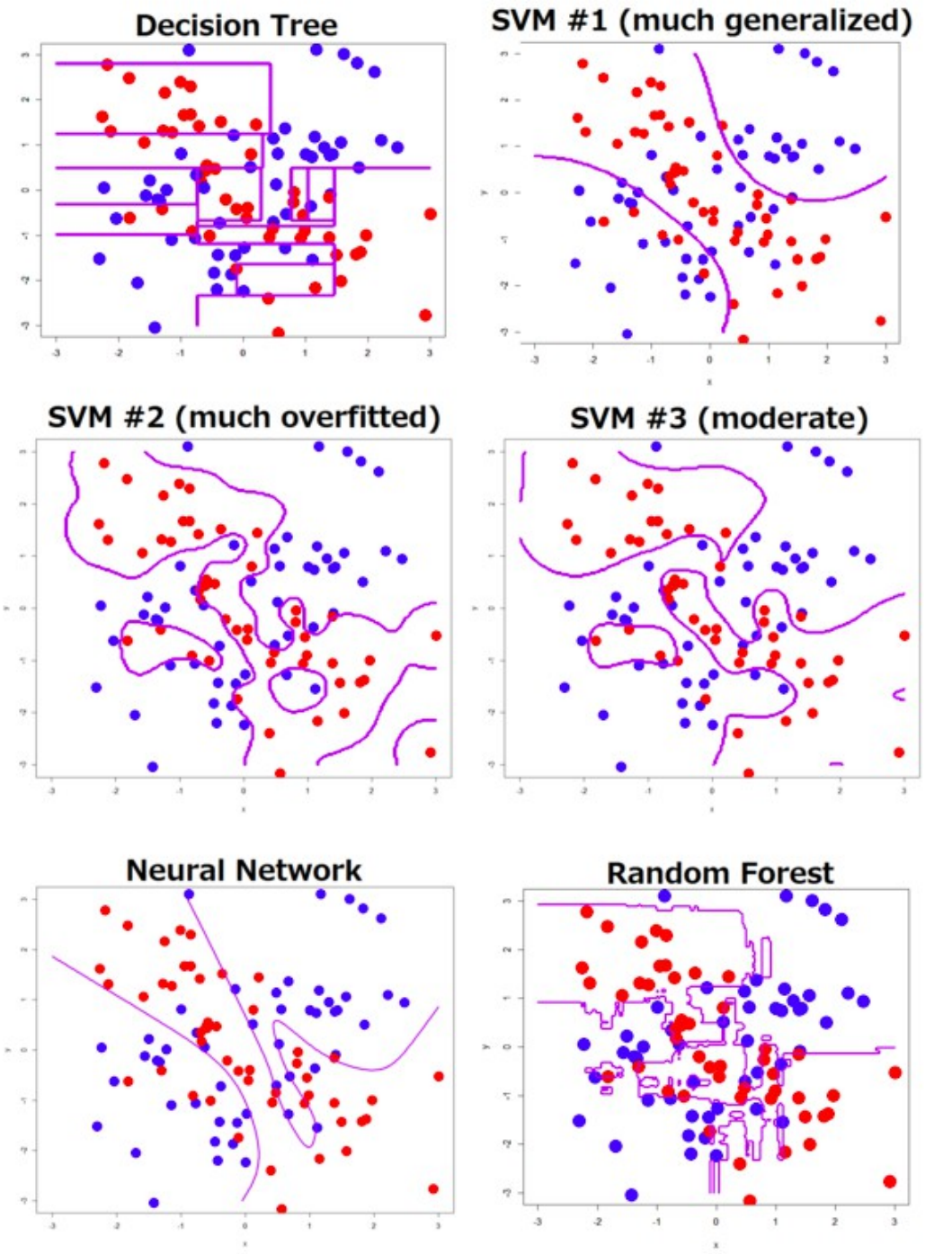
Etc.

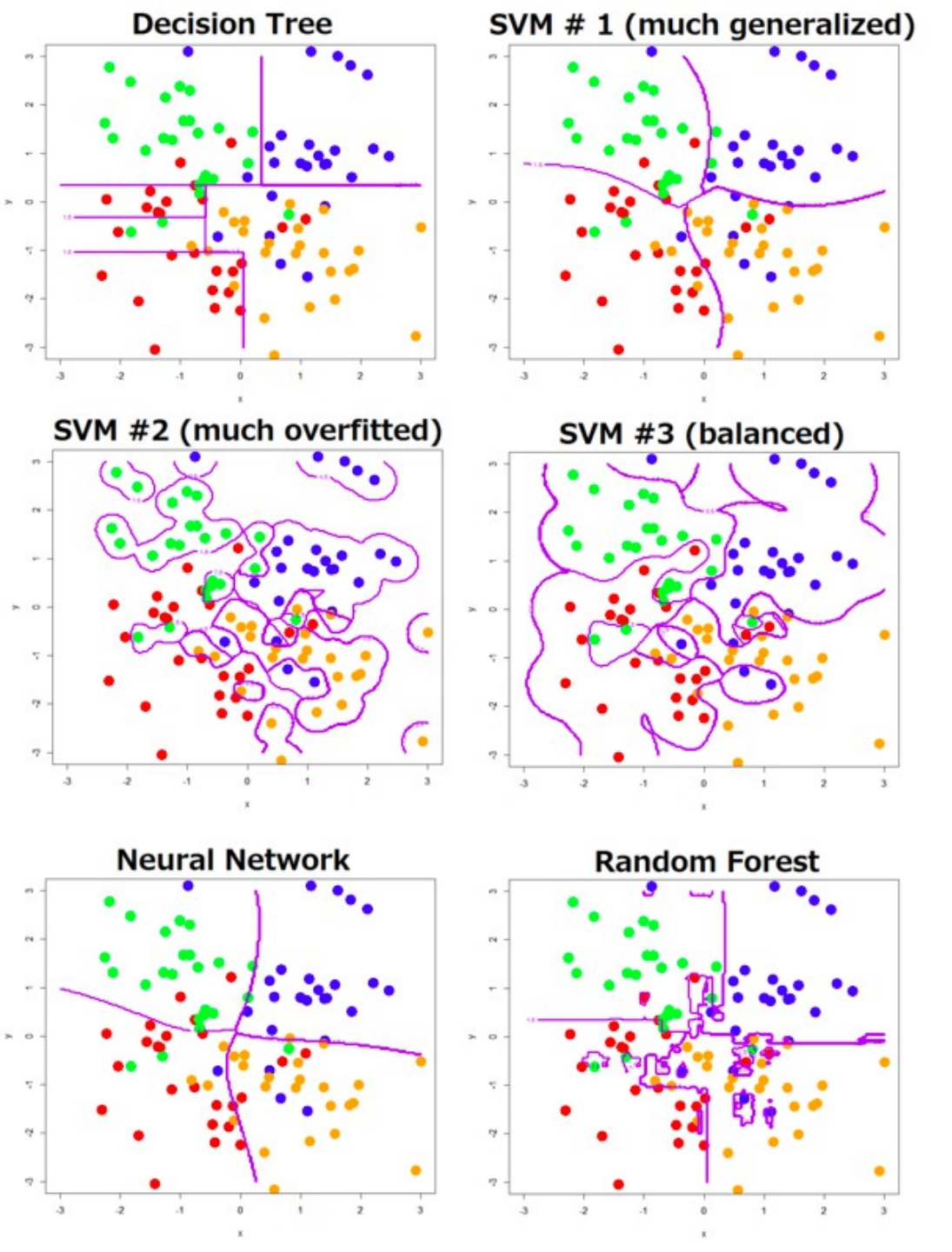


http://tjo-en.hatenablog.com/entry/2014/01/06/234155


Decision Tree
Multinomial Logit
SVM (Gaussian kernel)
Neural Network
Random Forest


Decision Tree
SVM (Gaussian kernel)
Neural Network
Random Forest


Decision Tree
SVM #1 (much generalized)
SVM #2 (much overfitted)
SVM #3 (moderate)
Neural Network
Random Forest


Decision Tree
SVM # 1 (much generalized)
SVM #2 (much overfitted)
SVM #3 (balanced)
Neural Network
Random Forest

What Modeler to Choose?

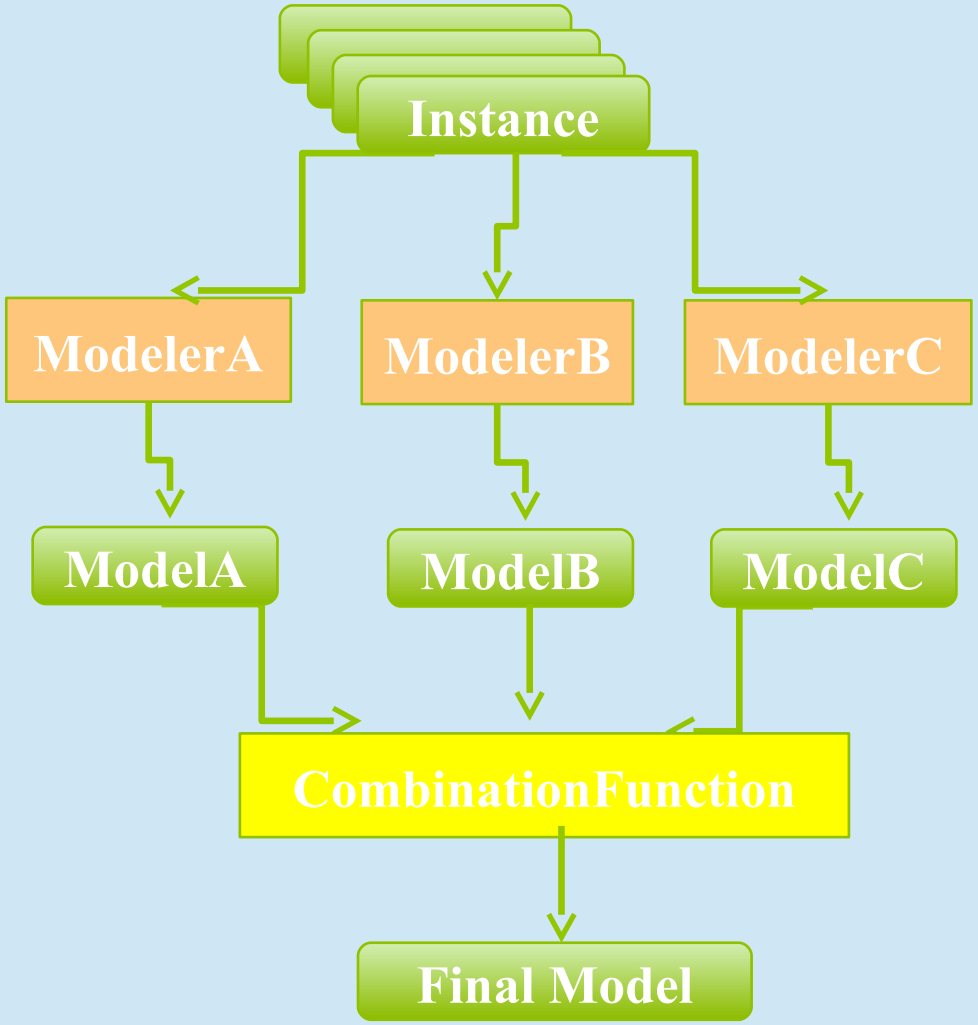
- Logistic regression
- Naïve Bayes classifiers
- Support vector machines (SVMs)
- Decision trees
- Random forests
- Kernel methods
- Genetic algorithms (GAs)
- Neural networks: perceptrons

Data scientists try different modelers, with different parameters, and check the accuracy to figure out which one works best for the data at hand

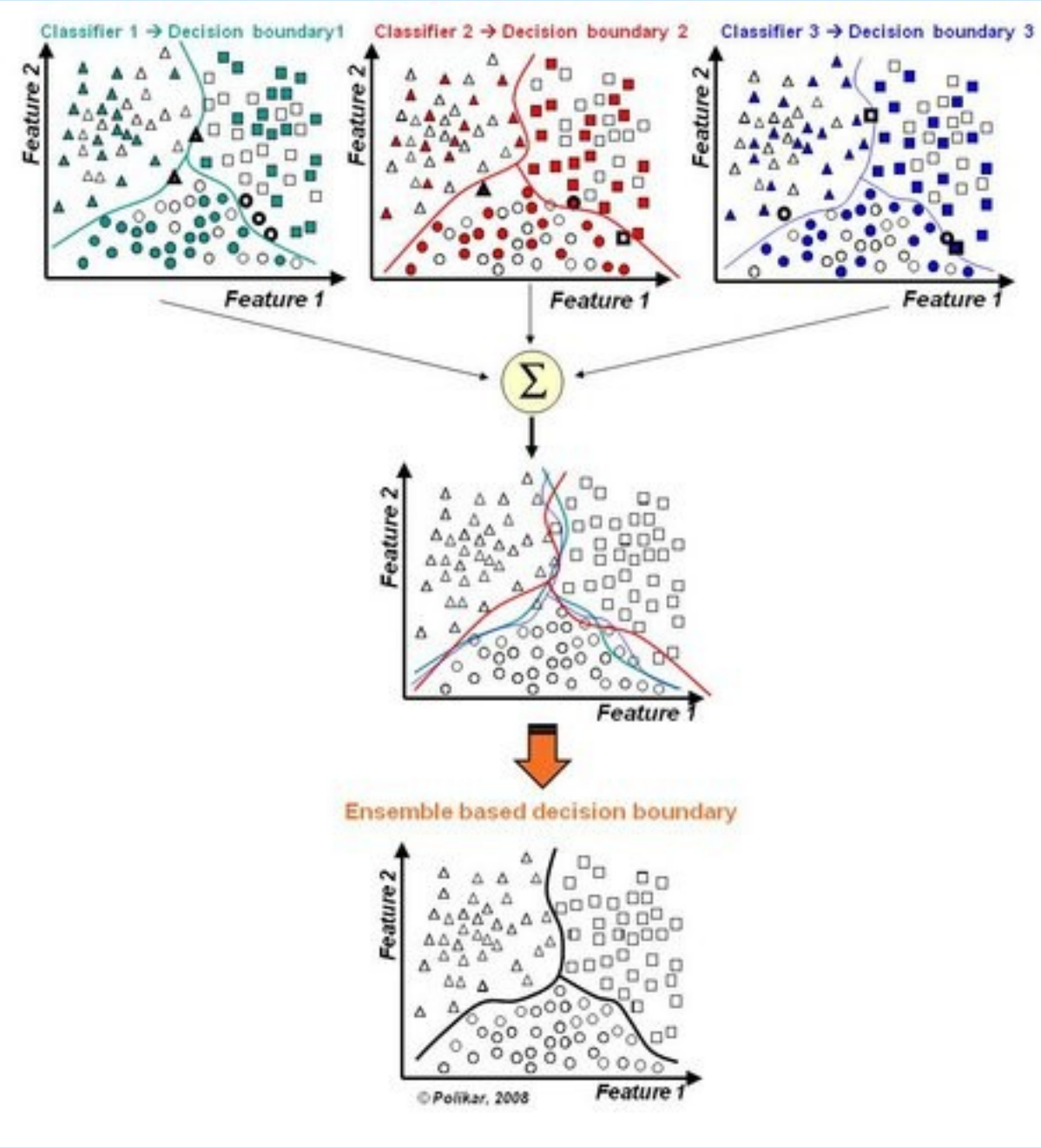
The loan data (reproduced)

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	**No**
2	young	false	false	good	**No**
3	young	true	false	good	**Yes**
4	young	true	true	fair	**Yes**
5	young	false	false	fair	**No**
6	middle	false	false	fair	**No**
7	middle	false	false	good	**No**
8	middle	true	true	good	**Yes**
9	middle	false	true	excellent	**Yes**
10	middle	false	true	excellent	**Yes**
11	old	false	true	excellent	**Yes**
12	old	false	true	good	**Yes**
13	old	true	false	good	**Yes**
14	old	true	false	excellent	**Yes**
15	old	false	false	fair	**No**

Ensembles



- An ensemble method uses several algorithms that do the same task, and combines their results
  - "Ensemble learning"
- A combination function joins the results
  - Majority vote: each algorithm gets a vote
  - Weighted voting: each algorithm's vote has a weight
  - Other complex combination functions



http://magizbox.com/index.php/machine-learning/ds-model-building/ensemble/

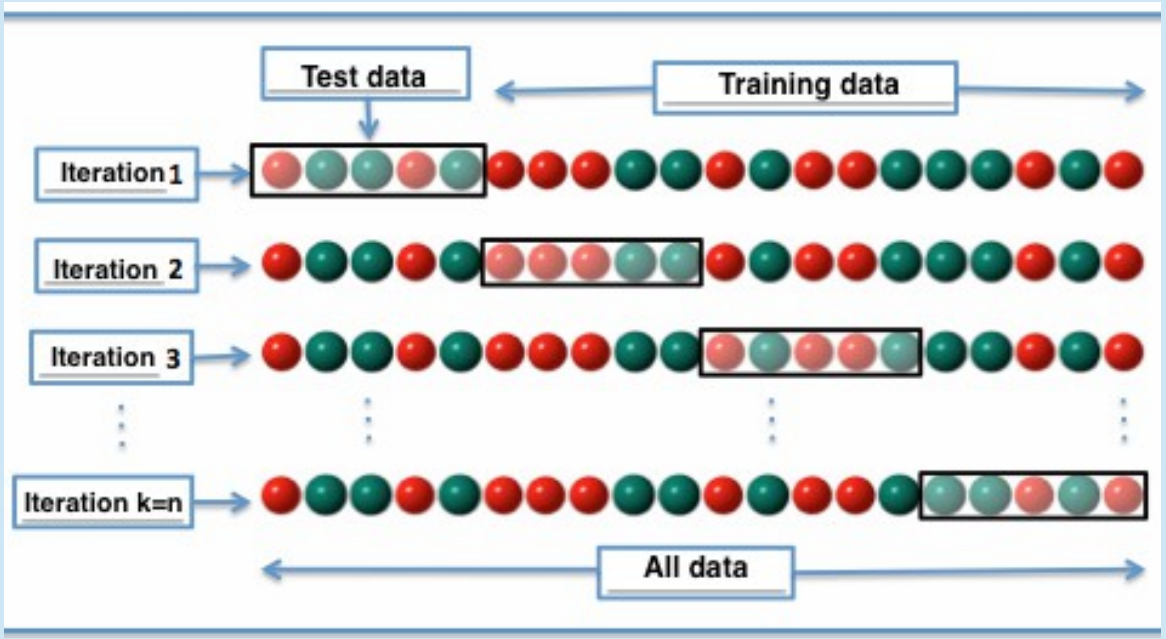
Classification Accuracy

- Accuracy: percentage of correct classifications

$$\text{Accuracy} = \frac{\text{Total test instances classified correctly}}{\text{Total number of test instances}}$$

Evaluating a Classifier: n-fold Cross Validation

- Suppose m labeled instances
  - Divide into n subsets ("*folds*") of equal size
- Run classifier n times, with each of the subsets as the test set
  - The rest (n-1) for training
  - Each run gives an accuracy result



Translated from image by Joan.domenech91 (Own work) [CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons (https://commons.wikimedia.org/wiki/File:K-fold_cross_validation.jpg)

Evaluating a Classifier: Confusion Matrix

	Classified positive	Classified negative
Actual positive	**True positive**	**False negative**
Actual negative	**False positive**	**True negative**

TP: number of positive examples classified correctly
FN: number of positive examples classified incorrectly
FP: number of negative examples classified incorrectly
TN: number of negative examples classified correctly

Evaluating a Classifier: Precision and Recall

TP: number of positive examples classified correctly
FN: number of positive examples classified incorrectly
FP: number of negative examples classified incorrectly
TN: number of negative examples classified correctly

$$\textbf{Precision} = \frac{TP}{TP + FP} \qquad \textbf{Recall} = \frac{TP}{TP + FN}$$

Note that the focus is on the positive class

Evaluating a Classifier: Other Metrics

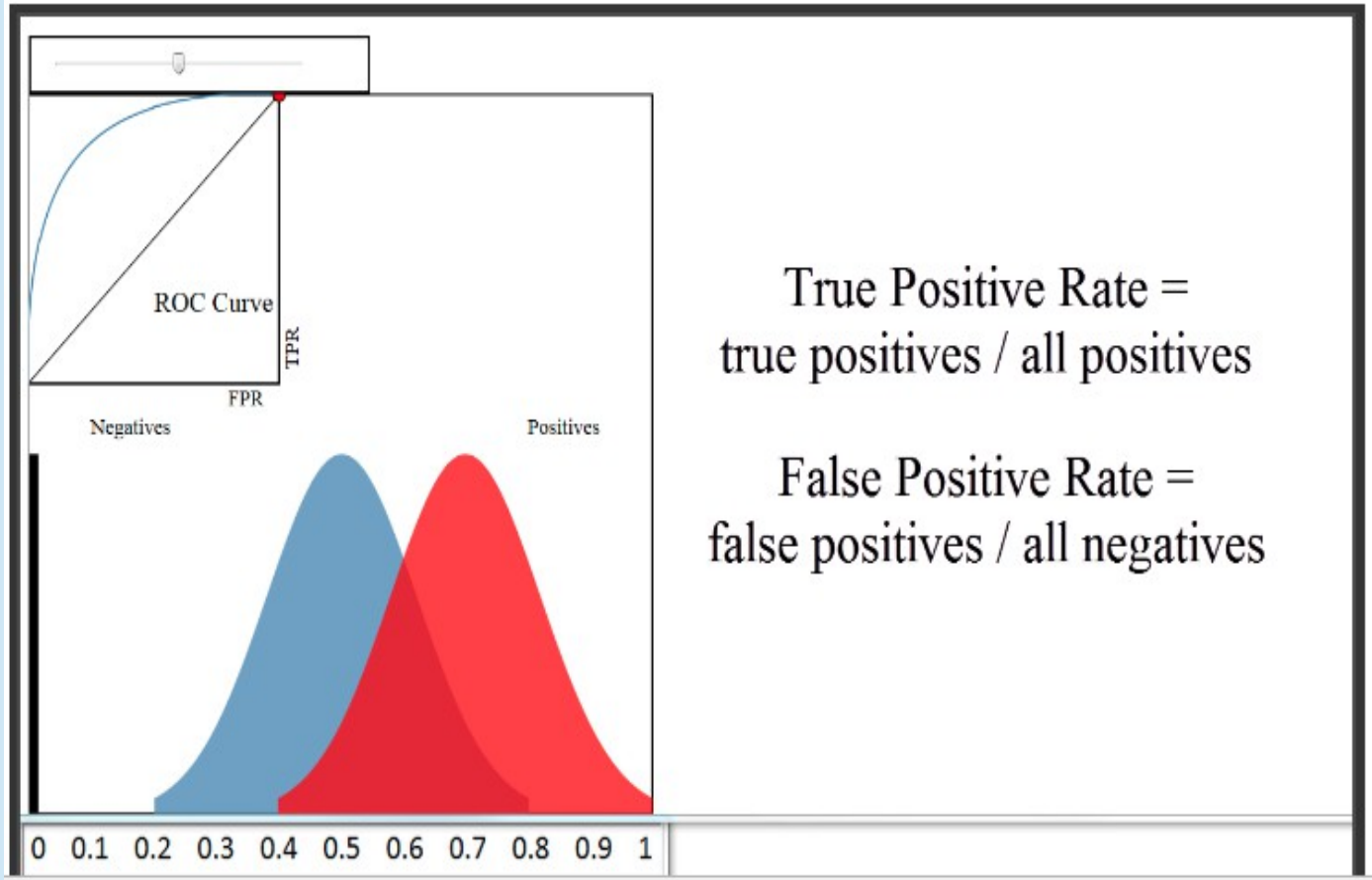
There are many other accuracy metrics

- F1-score
- Receive Operating Characteristics (ROC) curve
- Area Under the Curve (AUC)

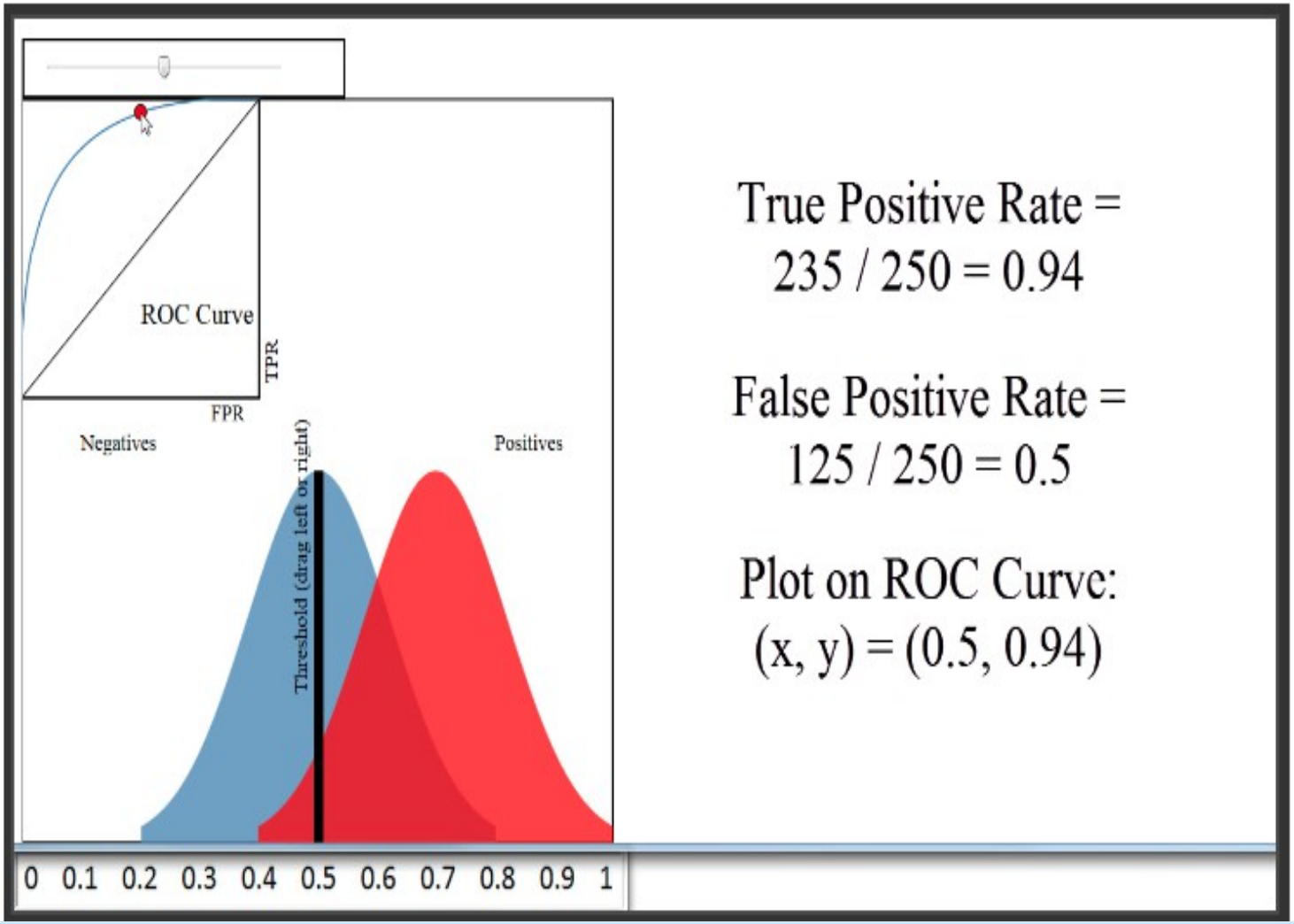
F1 score

In statistical analysis of binary classification, the F1 score (also F-score or F-measure) is a measure of a test's accuracy. It considers both the precision p and the recall r of the test to compute the score: p is the number of correct positive results divided by the number of all positive results returned by the classifier, and r is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive). The F1 score is the harmonic average of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0.

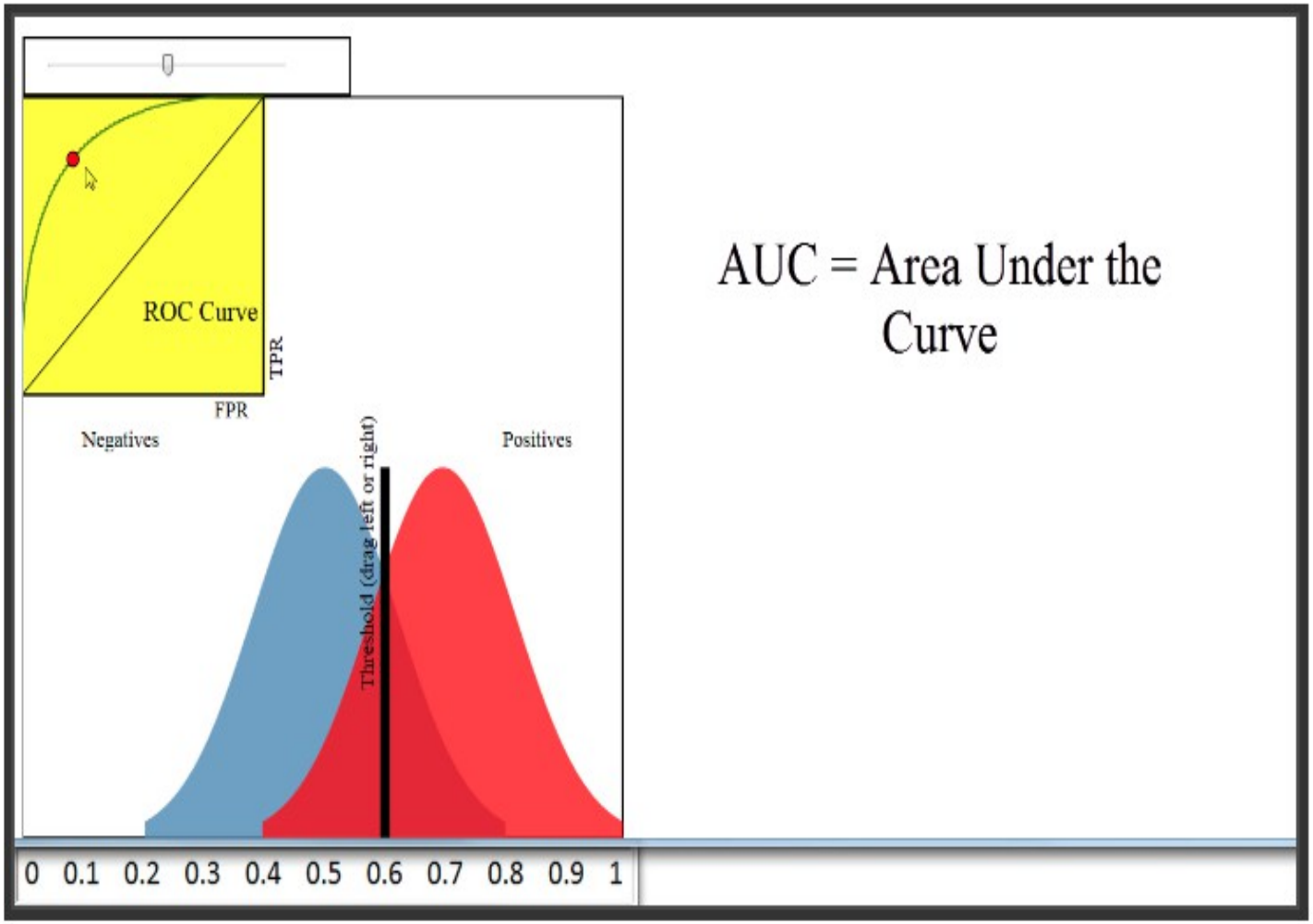
(**4:19**) Now let's talk about the ROC curve that you see here in the upper left. So, what is an ROC curve? It is a plot of the **True Positive Rate (on the y-axis)** versus the **False Positive Rate (on the x-axis)** for every possible classification threshold. As a **reminder**, the True Positive Rate answers the question, "When the actual classification is positive (meaning admitted), how often does the classfier predict positive?" The False Positive Rate answers the question, "When the actual classification is negative (meaning not admitted), how often does the classifier incorrectly predict positive?" Both the True Positive Rate and the False Positive Rate **range from 0 to 1**.



(6:16) Let's set a different threshold of 0.5. That would classify 360 papers as admitted, and 140 papers as not admitted. The True Positive Rate would be 235 divided by 250, or 0.94. The False Positive Rate would be 125 divided by 250, or 0.5. Thus, we would plot a point at 0.5 on the x-axis, and 0.94 on the y-axis, which is right here.



(8:55) Naturally, you might want to use the ROC curve to **quantify the performance of a classifier**, and give a higher score for this classifier than this classifier. That is the purpose of AUC, which stands for **Area Under the Curve**. AUC is literally just the percentage of this box that is under this curve. This classifier has an AUC of around 0.8, a very poor classifier has an AUC of around 0.5, and this classifier has an AUC of close to 1.



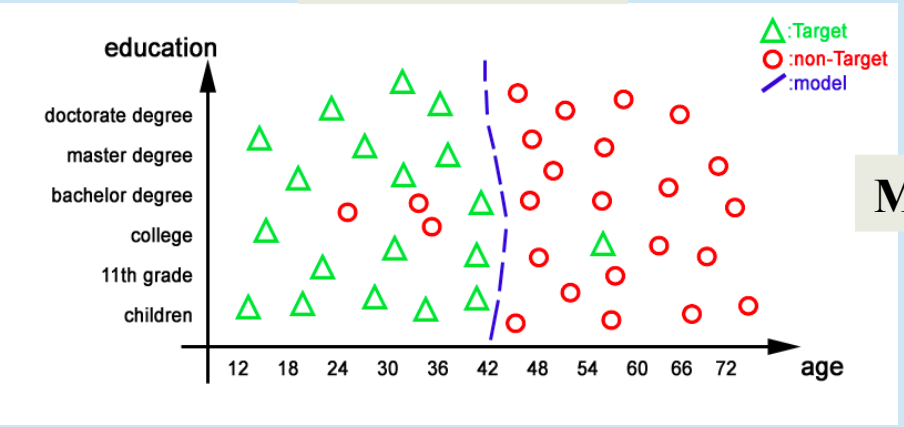
Evaluating a Classifier: What Affects the Performance

- Complexity of the task
  - Large amounts of features (high dimensionality)
  - Feature(s) appears very few times (sparse data)
- Few instances for a complex classification task
- Missing feature values for instances
- Errors in attribute values for instances

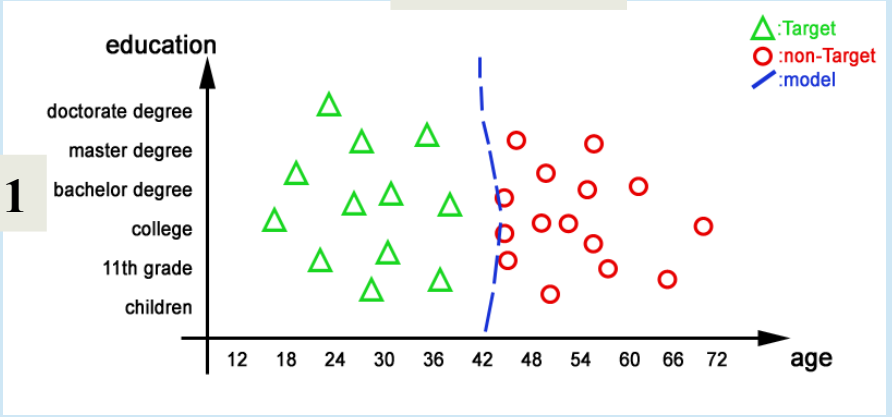
Overfitting

- A model overfits the training data when it is very accurate with that data, and may not do so well with new test data

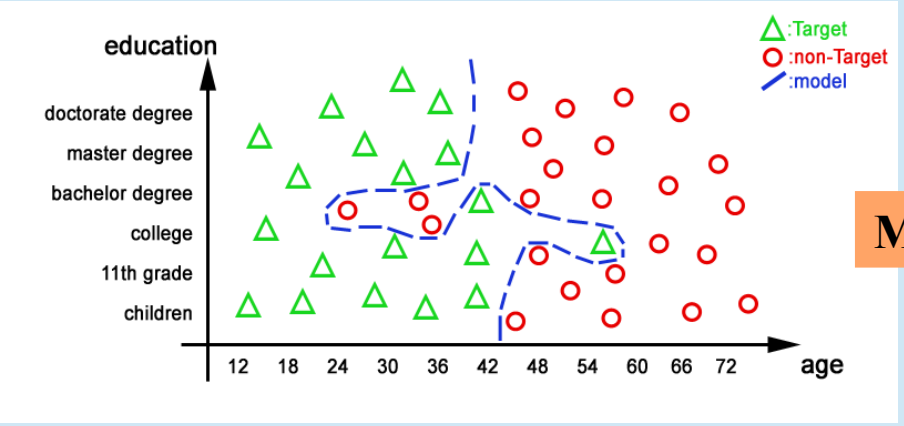
Training Data



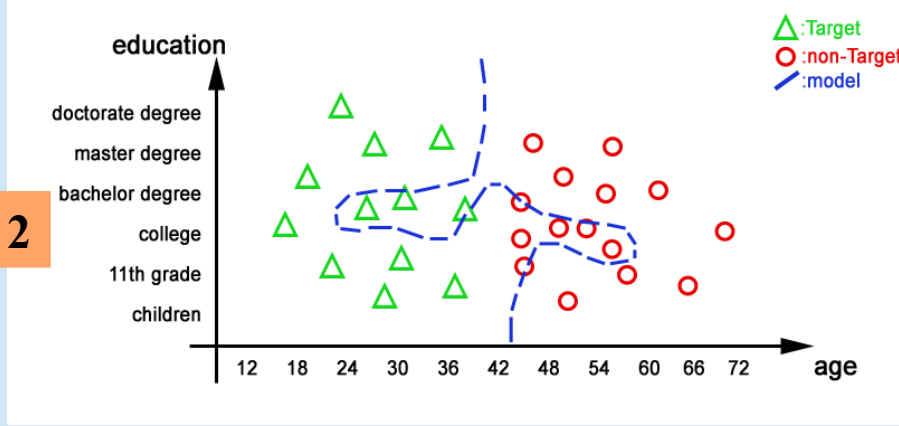
Test Data



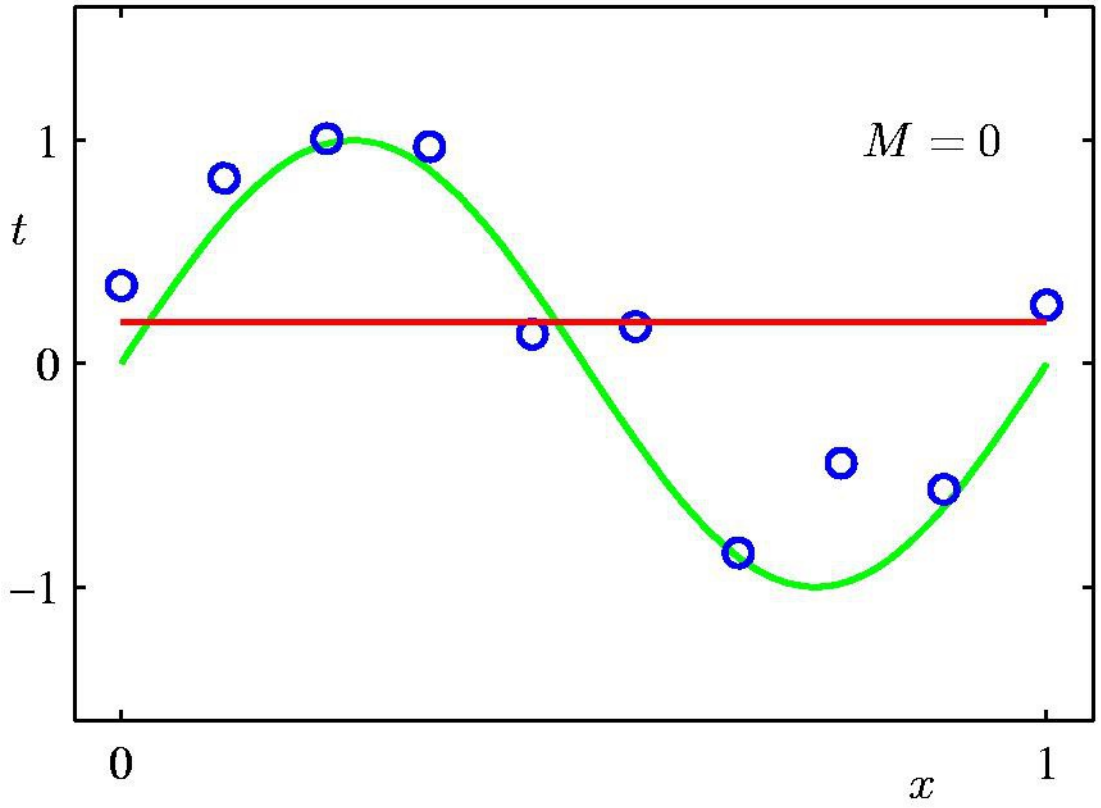
Model 1



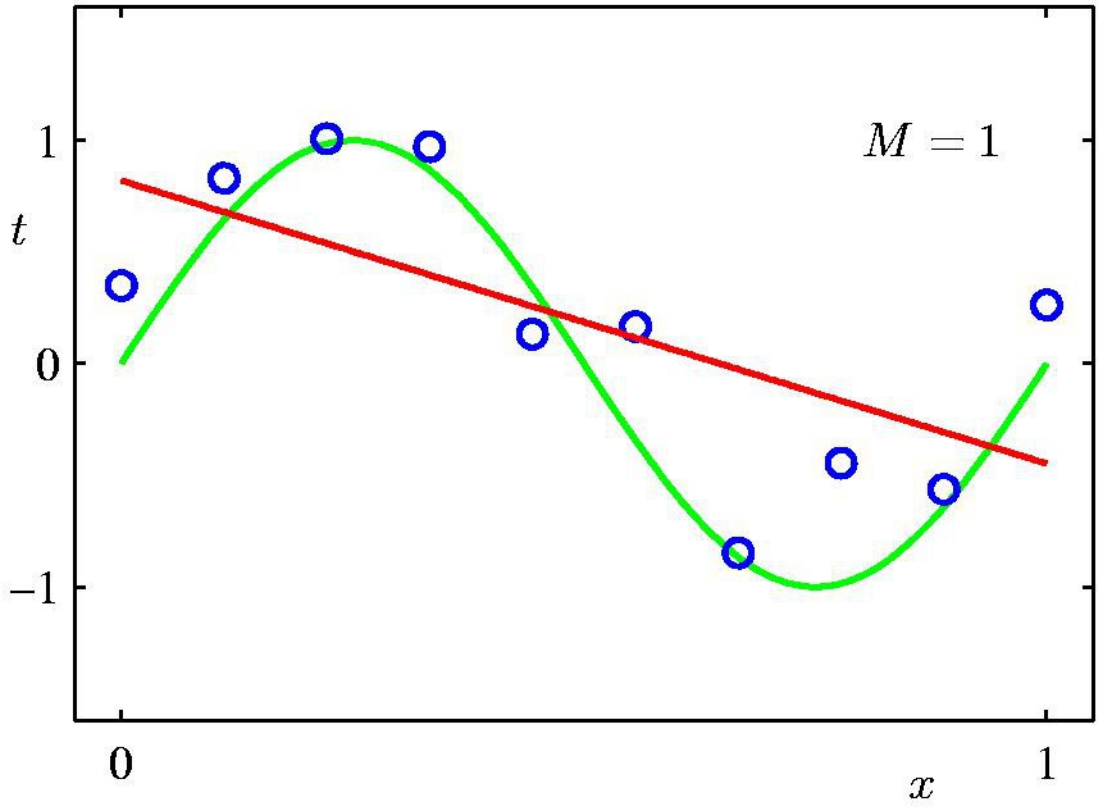
Model 2



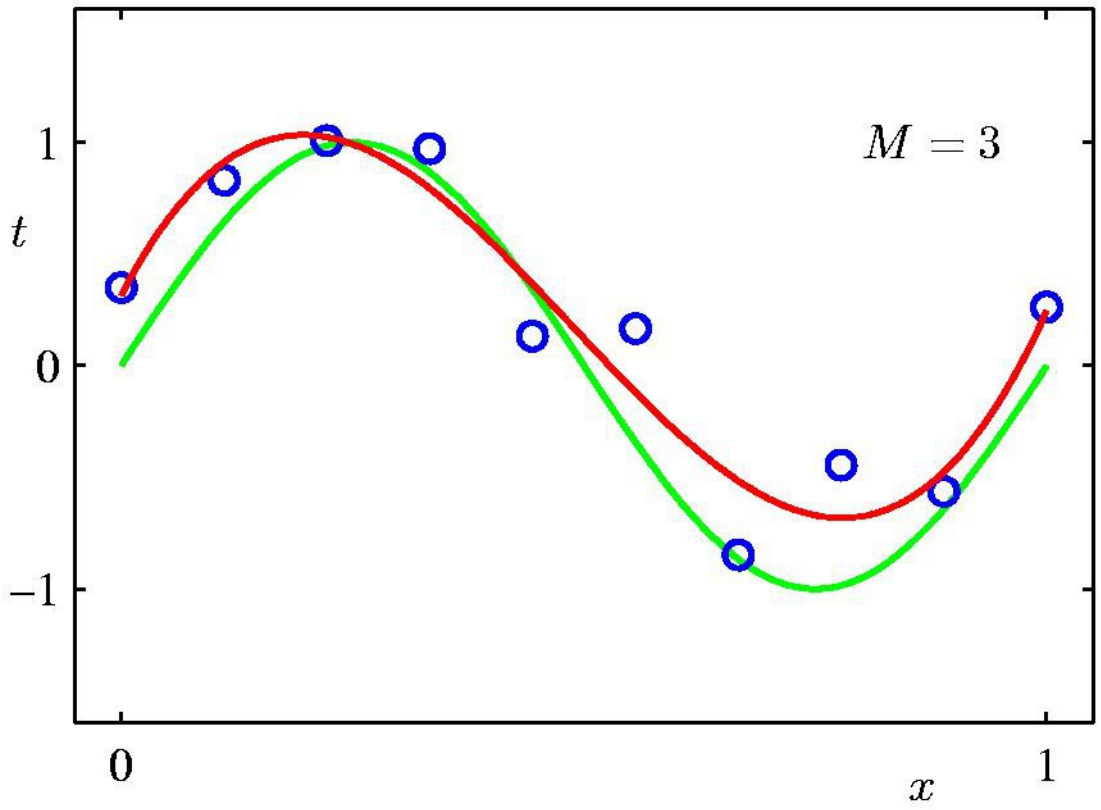
0th Order Polynomial



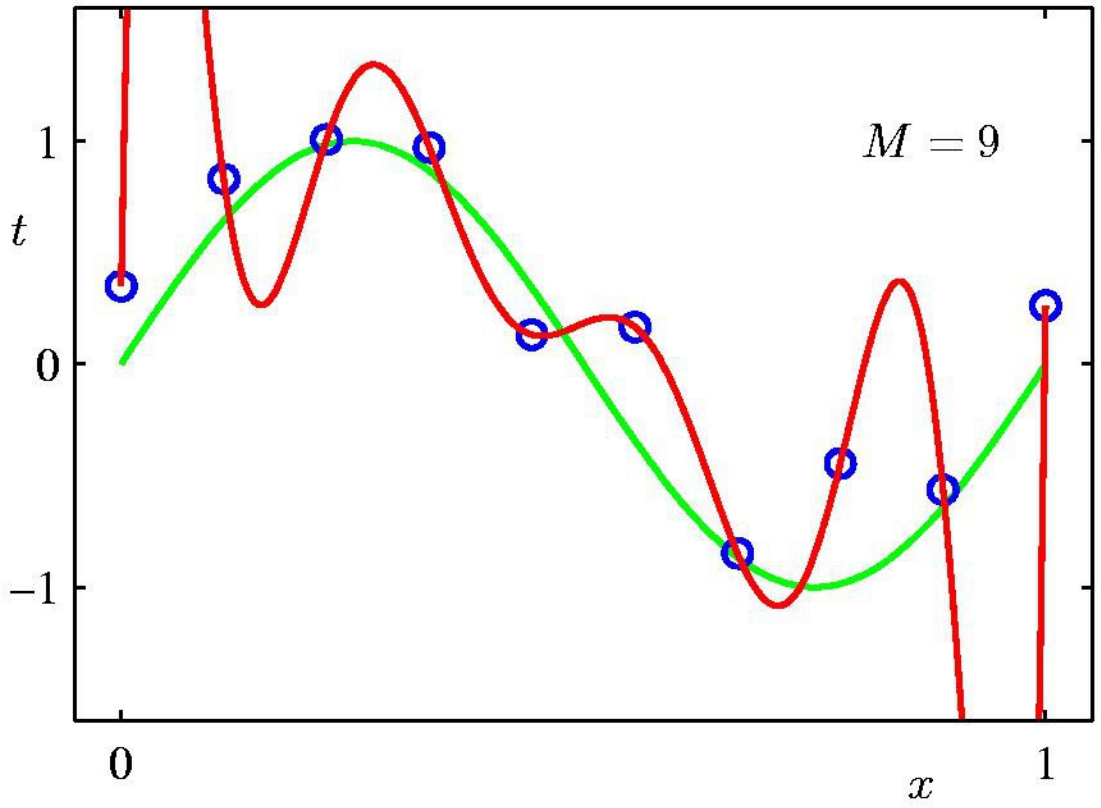
1st Order Polynomial



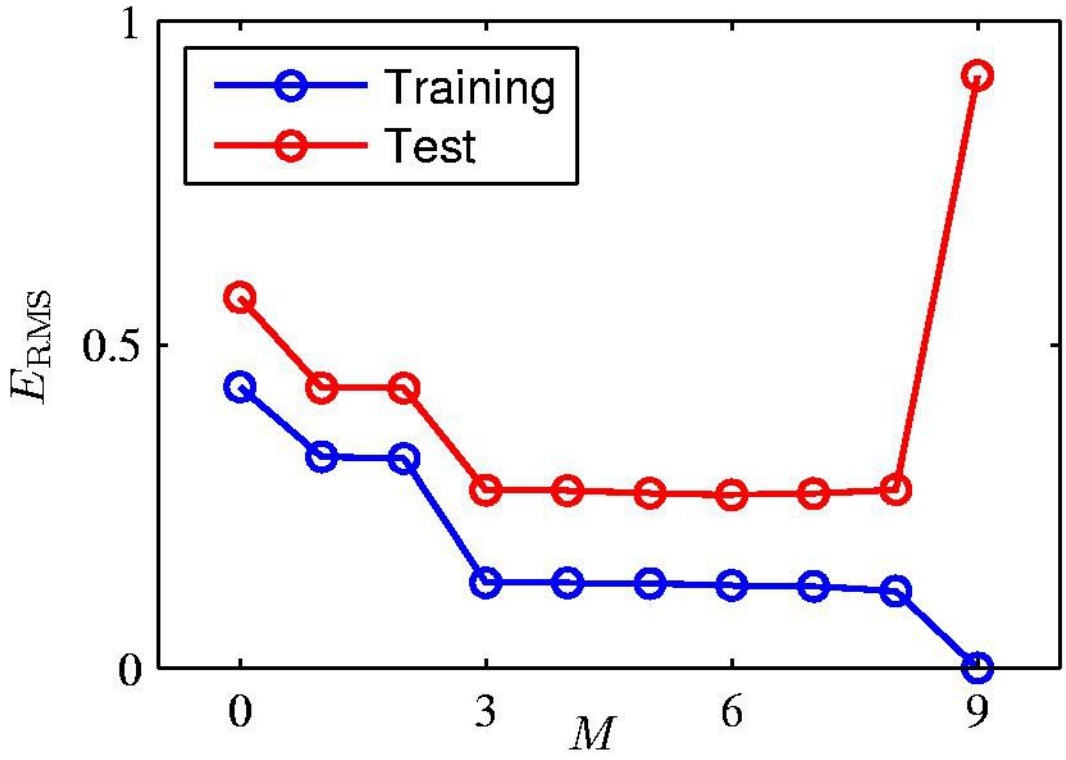
3rd Order Polynomial



9th Order Polynomial



Over-fitting

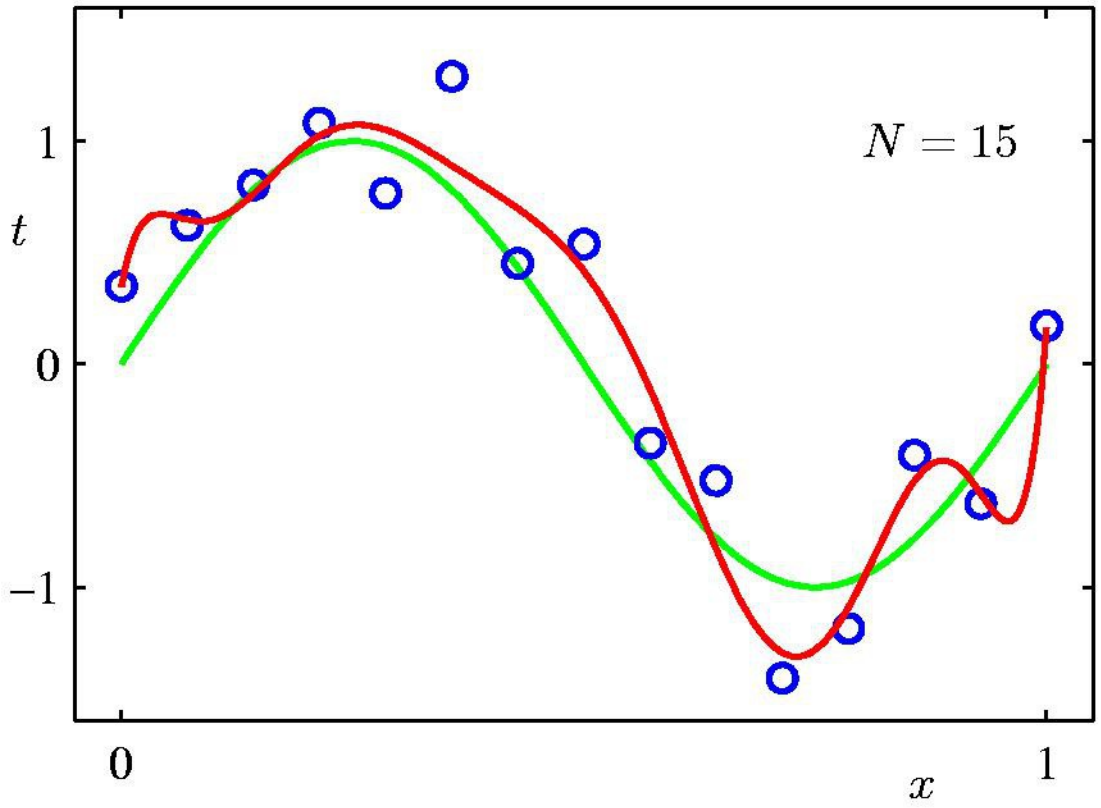


$$E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^{\star})/N}$$

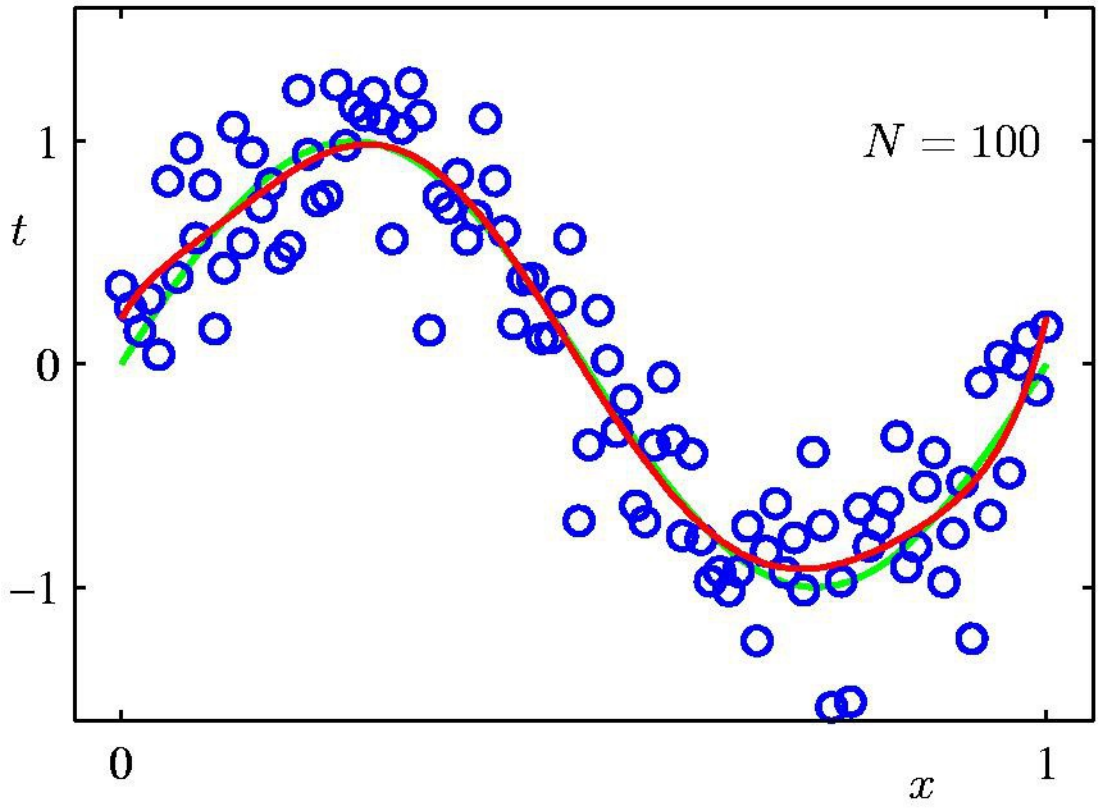
Polynomial Coefficients

	$M=0$	$M=1$	$M=3$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

Data Set Size: $N = 15$



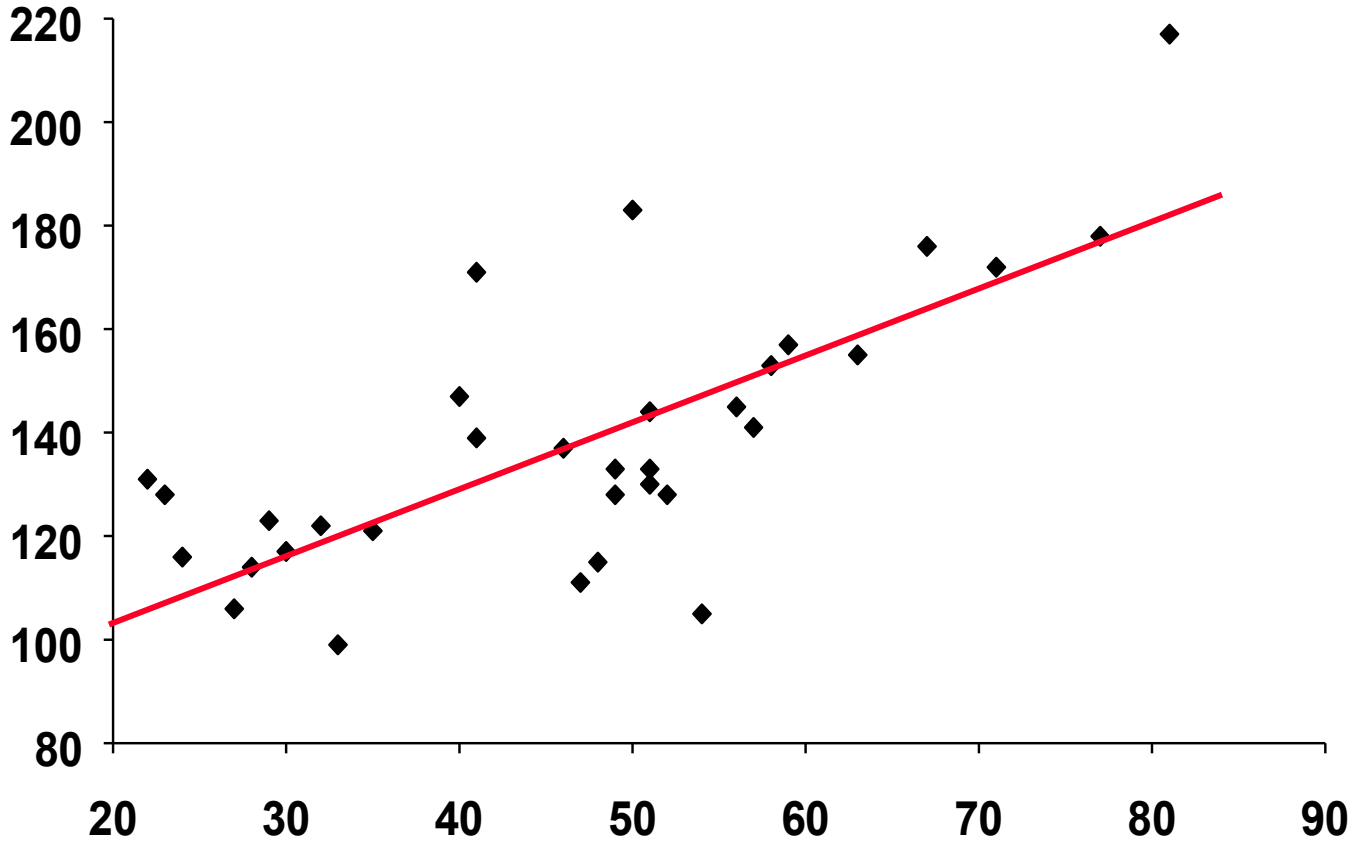
Data Set Size: $N = 100$



Simple linear regression

Table 1 Age and systolic blood pressure (SBP) among 33 adult women

Age	SBP	Age	SBP	Age	SBP
22	131	41	139	52	128
23	128	41	171	54	105
24	116	46	137	56	145
27	106	47	111	57	141
28	114	48	115	58	153
29	123	49	133	59	157
30	117	49	128	63	155
32	122	50	183	67	176
33	99	51	130	71	172
35	121	51	133	77	178
40	147	51	144	81	217



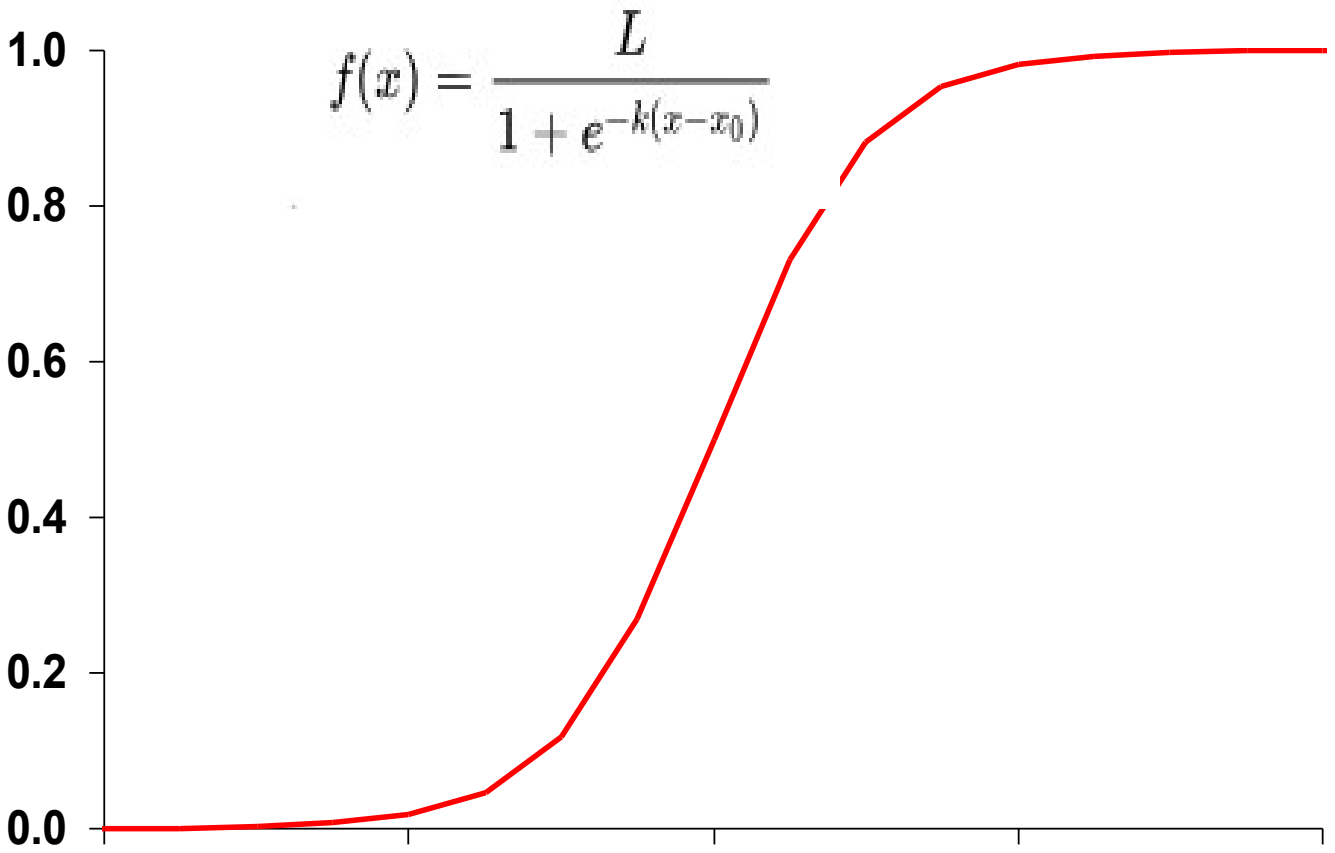
adapted from Colton T. Statistics in Medicine. Boston: Little Brown, 1974

Logistic regression (1)

Table 2 Age and signs of coronary heart disease (CD)

Age	CD	Age	CD	Age	CD
22	0	40	0	54	0
23	0	41	1	55	1
24	0	46	0	58	1
27	0	47	0	60	1
28	0	48	0	60	0
30	0	49	1	62	1
30	0	49	0	65	1
32	0	50	1	67	1
33	0	51	0	71	1
35	1	51	1	77	1
38	0	52	0	81	1

Logistic function (1)



Naive Bayes vs Logistic Regression

- **Generative model:** Naive Bayes

$$\hat{p}(x_i = 1|y = b) = \frac{s\{x_i = 1, y = b\} + l}{s\{y = b\} + 2l}$$

$$\hat{p}(y = b) = \frac{s\{y = b\}}{\sum_j s\{y = j\}}$$

- **Classify new example x based on ratio**

$$\frac{\hat{p}(y = T|x)}{\hat{p}(y = F|x)} = \frac{\hat{p}(y = T)\prod_{i=1}^{n}\hat{p}(x_i|y = T)}{\hat{p}(y = F)\prod_{i=1}^{n}\hat{p}(x_i|y = F)}$$

- **Logistic Regression:**

$$\hat{p}(y = T|x; \beta, \theta) = 1/(1 + exp(-\sum_{i=1}^{n}\beta_i x_i - \theta))$$

- **Recall: both classifiers are linear**

Rate of covergence: logistic regression

Proposition *Let h_{Dis} be logistic regression in n-dimensions. Then with high probability*

$$\varepsilon(h_{\mathrm{Dis}}) \leq \varepsilon(h_{\mathrm{Dis},\infty}) + O\left(\sqrt{\tfrac{n}{m}\log\tfrac{m}{n}}\right)$$

Thus, for $\varepsilon(h_{\mathrm{Dis}}) < \varepsilon(h_{\mathrm{Dis},\infty}) + \epsilon_0$ to hold with high probability (here, $\epsilon_0 > 0$ is some fixed constant), it suffices to pick $m = \Omega(n)$.

Convergences to best linear classifier, in order of n examples

follows from Vapnik's structural risk bound (VC-dimension of n dimensional linear separators is n+1)

Rate of covergence: Naive Bayes

We will proceed in 2 stages:

Consider how fast parameters converge to their optimal values

(we do not care about it, actually)

We care: Derive how it corresponds to the convergence of the error to the asymptotical error

The authors consider a continous case (where input is continious) but it is not very interesting for NLP

Convergence of Parameters

Lemma *Let any* $\epsilon_1, \delta > 0$ *and any* $l \geq 0$ *be fixed.*

Assume that for some fixed $\rho_0 > 0$, *we have that* $\rho_0 \leq p(y = T) \leq 1 - \rho_0$.

Let $m = O\left((1/\epsilon_1^2)\log(n/\delta)\right)$.

Then with probability at least $1 - \delta$:

$$|\hat{p}(x_i|y = b) - p(x_i|y = b)| \leq \epsilon_1$$
$$|\hat{p}(y = b) - p(y = b)| \leq \epsilon_1$$

for all $i = 1, \ldots, n$ *and* $b \in \mathcal{Y}$.

Recall: Chernoff Bound

- Consider a binary random variable X (e.g., the result of a coin toss) which has probability p of being *head (1)*
- Consider m samples, $\{x_1, x_2, \ldots\}$, drawn independently from X.
- the maximum likelihood estimate of p is the relative frequency of $x_i = 1$. That is

$$\hat{p} = \sum_i x_i / m.$$

- For all $p \in [0, 1], \epsilon > 0$,

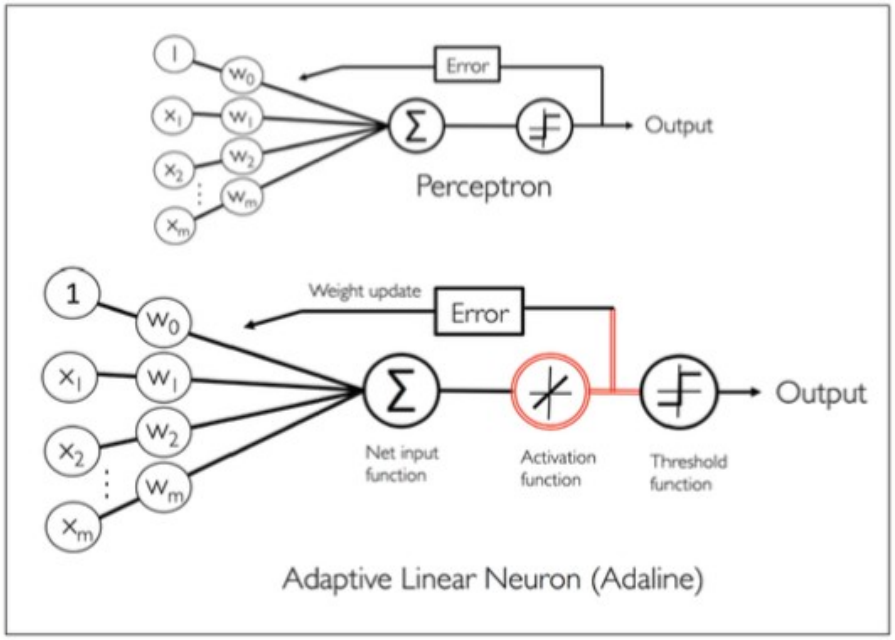
$$P[|p - \hat{p}| > \epsilon] \leq 2e^{-2m\epsilon^2}$$

Adaline, Perceptron and MLP

Adaline Rule Vs Perceptron Rule

Adaline Rule

- Weights are updated based on a linear activation function.
- Compares the true class labels with the linear activation function's continuous valued output to compute the model error and update the weights.



Perceptron Rule

- Weights are updated based on a unit step function.
- Compares the true class labels with the predicted class labels to compute the model error and update the weights.

Steps to Minimize Cost Functions With Gradient Descent

Using gradient descent, update weight by taking a step in the opposite direction of the gradient.

$$\boldsymbol{w} \coloneqq \boldsymbol{w} + \Delta \boldsymbol{w}$$

The weight change "Δw" is defined as negative gradient multiplied by the learning rate "η".

$$\Delta \boldsymbol{w} = -\eta \nabla J(\boldsymbol{w})$$

To compute the gradient of the cost function, compute the partial derivative of the cost function with respect to each weight "w_j". So it can be written as:

$$\frac{\partial J}{\partial w_j} = -\sum_i \left(y^{(i)} - \phi\left(z^{(i)}\right) \right) x_j^{(i)}$$

$$\Delta w_j = -\eta \frac{\partial J}{\partial w_j} = \eta \sum_i \left(y^{(i)} - \phi\left(z^{(i)}\right) \right) x_j^{(i)}$$

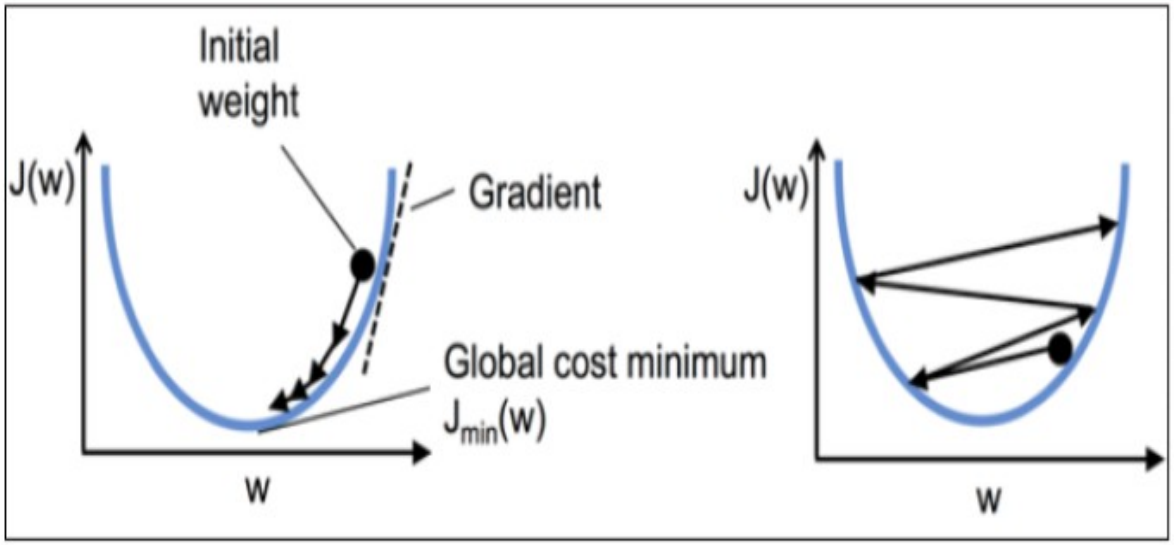
Since all weights are updated simultaneously, the Adaline learning rule becomes:

$$\boldsymbol{w} \coloneqq \boldsymbol{w} + \Delta \boldsymbol{w}$$

Tune the Learning Rate

CONVERGENCE

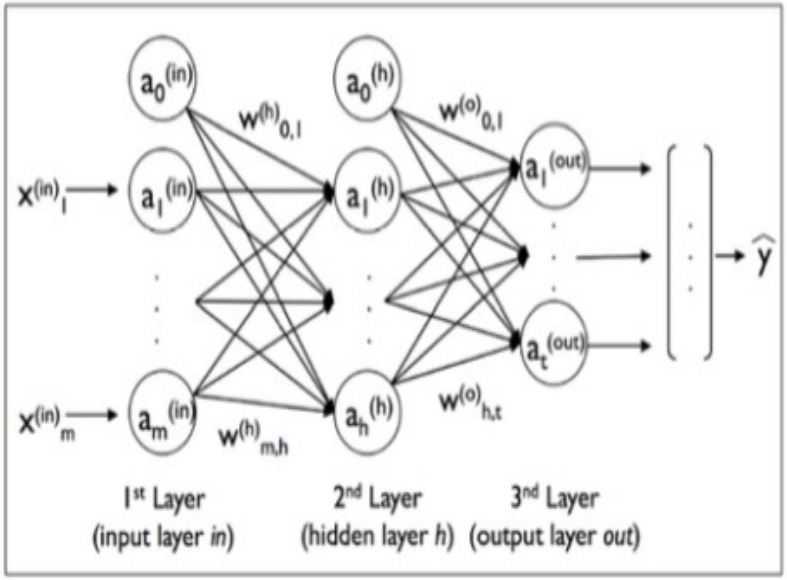
The figure on the left demonstrates optimum learning rate, where the cost function converges to a global minimum.



The figure on the right shows a large learning rate, and the global minimum gets missed during weight adjustments.

Multi-layer ANN

- A fully connected multi-layer neural network is called a **Multilayer Perceptron (MLP)**.
- It has 3 layers including one hidden layer.
- If it has more than 1 hidden layer, it is called a deep ANN.
- An MLP is a typical example of a feedforward artificial neural network.



In this figure, the i^{th} activation unit in the l^{th} layer is denoted as $a_i^{(l)}$.

Backpropagation in a Neural Network

CALCULATION STEPS

Calculate error vector of output layer.
Here, y is the vector of ground truth labels.

$$\delta^{(out)} = a^{(out)} - y$$

$$\delta^{(h)} = \delta^{(out)} \left(W^{(out)}\right)^T \odot \frac{\partial \phi\left(z^{(h)}\right)}{\partial z^{(h)}}$$

Calculate error term of the hidden layer. Last term is the derivative of the sigmoid activation function.

The derivative of the sigmoid activation can be shown to be equal to:

$$\frac{\partial \phi(z)}{\partial z} = \left(a^{(h)} \odot \left(1 - a^{(h)}\right)\right)$$

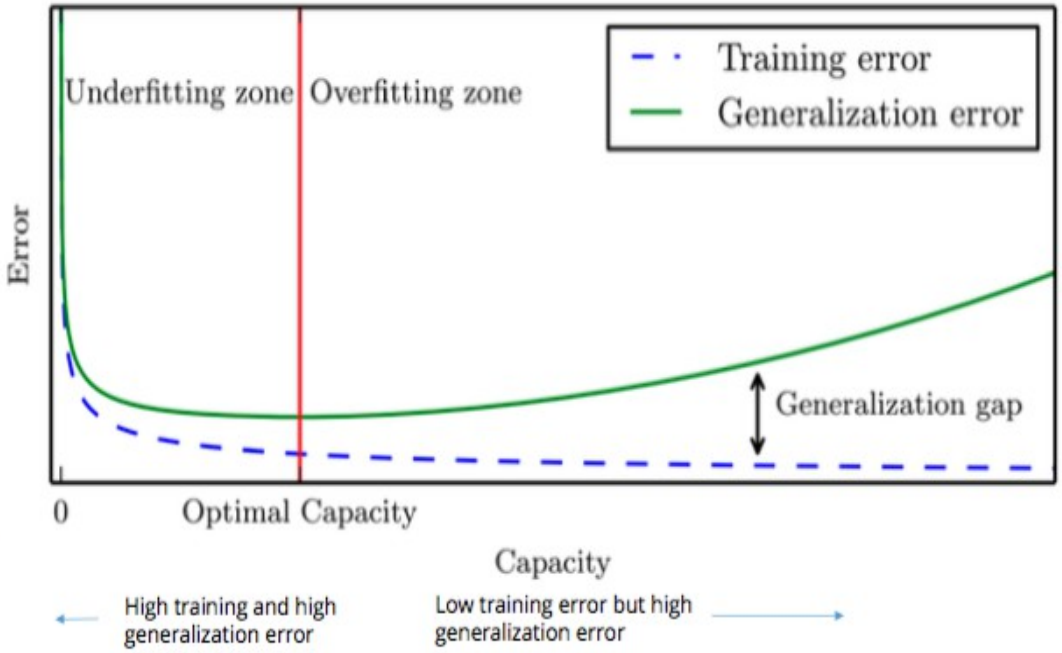
$$\delta^{(h)} = \delta^{(out)} \left(W^{(out)}\right)^T \odot \left(a^{(h)} \odot \left(1 - a^{(h)}\right)\right)$$

This can also be written as given.

Overfitting

GENERALIZATION AND CAPACITY

- At lower capacity, the model does not fit well. It is underfitted.
- At higher capacity, it might fit well but might tend to overfit, causing a large generalization error.
- An optimum point for the capacity is found between underfitting and overfitting zones, where generalization error is close to minimum.



Thank You!